Package: checkarg (via r-universe)

October 19, 2026

Title Check the Basic Validity of a (Function) Argument

Version 0.2.0

Description Utility functions that allow checking the basic validity of a function argument or any other value, including generating an error and assigning a default in a single line of code. The main purpose of the package is to provide simple and easily readable argument checking to improve code robustness.

Depends R (>= 3.1.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Repository https://ivwalle.r-universe.dev

Date/Publication 2018-07-02 21:34:59 UTC

RemoteUrl https://github.com/ivwalle/checkarg

RemoteRef HEAD

RemoteSha 32fd6d0f5cf7b302089796cd7f9db2dbc330baa1

Contents

checkarg 11
checkargFailedHandler 13
isBooleanOrNaScalar 14
isBooleanOrNaScalarOrNull 15
isBooleanOrNaVector 16
isBooleanOrNaVectorOrNull 17
isBooleanScalar 19
isBooleanScalarOrNull 20
isBooleanVector 21
isBooleanVectorOrNull 22
isIntegerOrInfScalar 24
isIntegerOrInfScalarOrNull 25

isIntegerOrInfVector . . . 26
isIntegerOrInfVectorOrNull . . . 27
isIntegerOrNanOrInfScalar . . . 29
isIntegerOrNanOrInfScalarOrNull . . . 30
isIntegerOrNanOrInfVector . . . 31
isIntegerOrNanOrInfVectorOrNull . . . 33
isIntegerOrNanScalar . . . 34
isIntegerOrNanScalarOrNull . . . 35
isIntegerOrNanVector . . . 36
isIntegerOrNanVectorOrNull . . . 38
isIntegerOrNaOrInfScalar . . . 39
isIntegerOrNaOrInfScalarOrNull . . . 40
isIntegerOrNaOrInfVector . . . 42
isIntegerOrNaOrInfVectorOrNull . . . 43
isIntegerOrNaOrNanOrInfScalar . . . 44
isIntegerOrNaOrNanOrInfScalarOrNull . . . 45
isIntegerOrNaOrNanOrInfVector . . . 47
isIntegerOrNaOrNanOrInfVectorOrNull . . . 48
isIntegerOrNaOrNanScalar . . . 49
isIntegerOrNaOrNanScalarOrNull . . . 51
isIntegerOrNaOrNanVector . . . 52
isIntegerOrNaOrNanVectorOrNull . . . 53
isIntegerOrNaScalar . . . 55
isIntegerOrNaScalarOrNull . . . 56
isIntegerOrNaVector . . . 57
isIntegerOrNaVectorOrNull . . . 58
isIntegerScalar . . . 60
isIntegerScalarOrNull . . . 61
isIntegerVector . . . 62
isIntegerVectorOrNull . . . 63
isNanScalar . . . 65
isNanScalarOrNull . . . 66
isNanVector . . . 67
isNanVectorOrNull . . . 68
isNaOrNanScalar . . . 69
isNaOrNanScalarOrNull . . . 70
isNaOrNanVector . . . 72
isNaOrNanVectorOrNull . . . 73
isNaOrNonEmptyStringScalar . . . 74
isNaOrNonEmptyStringScalarOrNull . . . 76
isNaOrNonEmptyStringVector . . . 77
isNaOrNonEmptyStringVectorOrNull . . . 78
isNaOrStringScalar . . . 79
isNaOrStringScalarOrNull . . . 81
isNaOrStringVector . . . 82
isNaOrStringVectorOrNull . . . 83
isNaScalar . . . 84
isNaScalarOrNull . . . 86

isNaVector . . . 87
isNaVectorOrNull . . . 88
isNegativeIntegerOrInfScalar . . . 89
isNegativeIntegerOrInfScalarOrNull . . . 90
isNegativeIntegerOrInfVector . . . 92
isNegativeIntegerOrInfVectorOrNull . . . 93
isNegativeIntegerOrNanOrInfScalar . . . 94
isNegativeIntegerOrNanOrInfScalarOrNull . . . 95
isNegativeIntegerOrNanOrInfVector . . . 97
isNegativeIntegerOrNanOrInfVectorOrNull . . . 98
isNegativeIntegerOrNanScalar . . . 99
isNegativeIntegerOrNanScalarOrNull . . . 101
isNegativeIntegerOrNanVector . . . 102
isNegativeIntegerOrNanVectorOrNull . . . 103
isNegativeIntegerOrNaOrInfScalar . . . 105
isNegativeIntegerOrNaOrInfScalarOrNull . . . 106
isNegativeIntegerOrNaOrInfVector . . . 107
isNegativeIntegerOrNaOrInfVectorOrNull . . . 109
isNegativeIntegerOrNaOrNanOrInfScalar . . . 110
isNegativeIntegerOrNaOrNanOrInfScalarOrNull . . . 111
isNegativeIntegerOrNaOrNanOrInfVector . . . 112
isNegativeIntegerOrNaOrNanOrInfVectorOrNull . . . 114
isNegativeIntegerOrNaOrNanScalar . . . 115
isNegativeIntegerOrNaOrNanScalarOrNull . . . 116
isNegativeIntegerOrNaOrNanVector . . . 118
isNegativeIntegerOrNaOrNanVectorOrNull . . . 119
isNegativeIntegerOrNaScalar . . . 120
isNegativeIntegerOrNaScalarOrNull . . . 121
isNegativeIntegerOrNaVector . . . 123
isNegativeIntegerOrNaVectorOrNull . . . 124
isNegativeIntegerScalar . . . 125
isNegativeIntegerScalarOrNull . . . 127
isNegativeIntegerVector . . . 128
isNegativeIntegerVectorOrNull . . . 129
isNegativeNumberOrInfScalar . . . 131
isNegativeNumberOrInfScalarOrNull . . . 132
isNegativeNumberOrInfVector . . . 133
isNegativeNumberOrInfVectorOrNull . . . 135
isNegativeNumberOrNanOrInfScalar . . . 136
isNegativeNumberOrNanOrInfScalarOrNull . . . 137
isNegativeNumberOrNanOrInfVector . . . 138
isNegativeNumberOrNanOrInfVectorOrNull . . . 140
isNegativeNumberOrNanScalar . . . 141
isNegativeNumberOrNanScalarOrNull . . . 142
isNegativeNumberOrNanVector . . . 144
isNegativeNumberOrNanVectorOrNull . . . 145
isNegativeNumberOrNaOrInfScalar . . . 146
isNegativeNumberOrNaOrInfScalarOrNull . . . 147

isNegativeNumberOrNaOrInfVector . . . 149
isNegativeNumberOrNaOrInfVectorOrNull . . . 150
isNegativeNumberOrNaOrNanOrInfScalar . . . 151
isNegativeNumberOrNaOrNanOrInfScalarOrNull . . . 153
isNegativeNumberOrNaOrNanOrInfVector . . . 154
isNegativeNumberOrNaOrNanOrInfVectorOrNull . . . 155
isNegativeNumberOrNaOrNanScalar . . . 157
isNegativeNumberOrNaOrNanScalarOrNull . . . 158
isNegativeNumberOrNaOrNanVector . . . 159
isNegativeNumberOrNaOrNanVectorOrNull . . . 161
isNegativeNumberOrNaScalar . . . 162
isNegativeNumberOrNaScalarOrNull . . . 163
isNegativeNumberOrNaVector . . . 164
isNegativeNumberOrNaVectorOrNull . . . 166
isNegativeNumberScalar . . . 167
isNegativeNumberScalarOrNull . . . 168
isNegativeNumberVector . . . 170
isNegativeNumberVectorOrNull . . . 171
isNonEmptyStringScalar . . . 172
isNonEmptyStringScalarOrNull . . . 173
isNonEmptyStringVector . . . 175
isNonEmptyStringVectorOrNull . . . 176
isNonZeroIntegerOrInfScalar . . . 177
isNonZeroIntegerOrInfScalarOrNull . . . 179
isNonZeroIntegerOrInfVector . . . 180
isNonZeroIntegerOrInfVectorOrNull . . . 181
isNonZeroIntegerOrNanOrInfScalar . . . 183
isNonZeroIntegerOrNanOrInfScalarOrNull . . . 184
isNonZeroIntegerOrNanOrInfVector . . . 185
isNonZeroIntegerOrNanOrInfVectorOrNull . . . 187
isNonZeroIntegerOrNanScalar . . . 188
isNonZeroIntegerOrNanScalarOrNull . . . 189
isNonZeroIntegerOrNanVector . . . 190
isNonZeroIntegerOrNanVectorOrNull . . . 192
isNonZeroIntegerOrNaOrInfScalar . . . 193
isNonZeroIntegerOrNaOrInfScalarOrNull . . . 194
isNonZeroIntegerOrNaOrInfVector . . . 196
isNonZeroIntegerOrNaOrInfVectorOrNull . . . 197
isNonZeroIntegerOrNaOrNanOrInfScalar . . . 198
isNonZeroIntegerOrNaOrNanOrInfScalarOrNull . . . 199
isNonZeroIntegerOrNaOrNanOrInfVector . . . 201
isNonZeroIntegerOrNaOrNanOrInfVectorOrNull . . . 202
isNonZeroIntegerOrNaOrNanScalar . . . 203
isNonZeroIntegerOrNaOrNanScalarOrNull . . . 205
isNonZeroIntegerOrNaOrNanVector . . . 206
isNonZeroIntegerOrNaOrNanVectorOrNull . . . 207
isNonZeroIntegerOrNaScalar . . . 209
isNonZeroIntegerOrNaScalarOrNull . . . 210

isNonZeroIntegerOrNaVector . . . 211
isNonZeroIntegerOrNaVectorOrNull . . . 213
isNonZeroIntegerScalar . . . 214
isNonZeroIntegerScalarOrNull . . . 215
isNonZeroIntegerVector . . . 216
isNonZeroIntegerVectorOrNull . . . 218
isNonZeroNumberOrInfScalar . . . 219
isNonZeroNumberOrInfScalarOrNull . . . 220
isNonZeroNumberOrInfVector . . . 222
isNonZeroNumberOrInfVectorOrNull . . . 223
isNonZeroNumberOrNanOrInfScalar . . . 224
isNonZeroNumberOrNanOrInfScalarOrNull . . . 225
isNonZeroNumberOrNanOrInfVector . . . 227
isNonZeroNumberOrNanOrInfVectorOrNull . . . 228
isNonZeroNumberOrNanScalar . . . 229
isNonZeroNumberOrNanScalarOrNull . . . 231
isNonZeroNumberOrNanVector . . . 232
isNonZeroNumberOrNanVectorOrNull . . . 233
isNonZeroNumberOrNaOrInfScalar . . . 235
isNonZeroNumberOrNaOrInfScalarOrNull . . . 236
isNonZeroNumberOrNaOrInfVector . . . 237
isNonZeroNumberOrNaOrInfVectorOrNull . . . 239
isNonZeroNumberOrNaOrNanOrInfScalar . . . 240
isNonZeroNumberOrNaOrNanOrInfScalarOrNull . . . 241
isNonZeroNumberOrNaOrNanOrInfVector . . . 242
isNonZeroNumberOrNaOrNanOrInfVectorOrNull . . . 244
isNonZeroNumberOrNaOrNanScalar . . . 245
isNonZeroNumberOrNaOrNanScalarOrNull . . . 246
isNonZeroNumberOrNaOrNanVector . . . 248
isNonZeroNumberOrNaOrNanVectorOrNull . . . 249
isNonZeroNumberOrNaScalar . . . 250
isNonZeroNumberOrNaScalarOrNull . . . 251
isNonZeroNumberOrNaVector . . . 253
isNonZeroNumberOrNaVectorOrNull . . . 254
isNonZeroNumberScalar . . . 255
isNonZeroNumberScalarOrNull . . . 257
isNonZeroNumberVector . . . 258
isNonZeroNumberVectorOrNull . . . 259
isNumberOrInfScalar . . . 261
isNumberOrInfScalarOrNull . . . 262
isNumberOrInfVector . . . 263
isNumberOrInfVectorOrNull . . . 264
isNumberOrNanOrInfScalar . . . 266
isNumberOrNanOrInfScalarOrNull . . . 267
isNumberOrNanOrInfVector . . . 268
isNumberOrNanOrInfVectorOrNull . . . 270
isNumberOrNanScalar . . . 271
isNumberOrNanScalarOrNull . . . 272

isNumberOrNanVector . . . 273
isNumberOrNanVectorOrNull . . . 275
isNumberOrNaOrInfScalar . . . 276
isNumberOrNaOrInfScalarOrNull . . . 277
isNumberOrNaOrInfVector . . . 279
isNumberOrNaOrInfVectorOrNull . . . 280
isNumberOrNaOrNanOrInfScalar . . . 281
isNumberOrNaOrNanOrInfScalarOrNull . . . 282
isNumberOrNaOrNanOrInfVector . . . 284
isNumberOrNaOrNanOrInfVectorOrNull . . . 285
isNumberOrNaOrNanScalar . . . 286
isNumberOrNaOrNanScalarOrNull . . . 288
isNumberOrNaOrNanVector . . . 289
isNumberOrNaOrNanVectorOrNull . . . 290
isNumberOrNaScalar . . . 292
isNumberOrNaScalarOrNull . . . 293
isNumberOrNaVector . . . 294
isNumberOrNaVectorOrNull . . . 295
isNumberScalar . . . 297
isNumberScalarOrNull . . . 298
isNumberVector . . . 299
isNumberVectorOrNull . . . 300
isPositiveIntegerOrInfScalar . . . 302
isPositiveIntegerOrInfScalarOrNull . . . 303
isPositiveIntegerOrInfVector . . . 304
isPositiveIntegerOrInfVectorOrNull . . . 306
isPositiveIntegerOrNanOrInfScalar . . . 307
isPositiveIntegerOrNanOrInfScalarOrNull . . . 308
isPositiveIntegerOrNanOrInfVector . . . 309
isPositiveIntegerOrNanOrInfVectorOrNull . . . 311
isPositiveIntegerOrNanScalar . . . 312
isPositiveIntegerOrNanScalarOrNull . . . 313
isPositiveIntegerOrNanVector . . . 315
isPositiveIntegerOrNanVectorOrNull . . . 316
isPositiveIntegerOrNaOrInfScalar . . . 317
isPositiveIntegerOrNaOrInfScalarOrNull . . . 318
isPositiveIntegerOrNaOrInfVector . . . 320
isPositiveIntegerOrNaOrInfVectorOrNull . . . 321
isPositiveIntegerOrNaOrNanOrInfScalar . . . 322
isPositiveIntegerOrNaOrNanOrInfScalarOrNull . . . 324
isPositiveIntegerOrNaOrNanOrInfVector . . . 325
isPositiveIntegerOrNaOrNanOrInfVectorOrNull . . . 326
isPositiveIntegerOrNaOrNanScalar . . . 328
isPositiveIntegerOrNaOrNanScalarOrNull . . . 329
isPositiveIntegerOrNaOrNanVector . . . 330
isPositiveIntegerOrNaOrNanVectorOrNull . . . 332
isPositiveIntegerOrNaScalar . . . 333
isPositiveIntegerOrNaScalarOrNull . . . 334

isPositiveIntegerOrNaVector . . . 335
isPositiveIntegerOrNaVectorOrNull . . . 337
isPositiveIntegerScalar . . . 338
isPositiveIntegerScalarOrNull . . . 339
isPositiveIntegerVector . . . 341
isPositiveIntegerVectorOrNull . . . 342
isPositiveNumberOrInfScalar . . . 343
isPositiveNumberOrInfScalarOrNull . . . 344
isPositiveNumberOrInfVector . . . 346
isPositiveNumberOrInfVectorOrNull . . . 347
isPositiveNumberOrNanOrInfScalar . . . 348
isPositiveNumberOrNanOrInfScalarOrNull . . . 350
isPositiveNumberOrNanOrInfVector . . . 351
isPositiveNumberOrNanOrInfVectorOrNull . . . 352
isPositiveNumberOrNanScalar . . . 354
isPositiveNumberOrNanScalarOrNull . . . 355
isPositiveNumberOrNanVector . . . 356
isPositiveNumberOrNanVectorOrNull . . . 358
isPositiveNumberOrNaOrInfScalar . . . 359
isPositiveNumberOrNaOrInfScalarOrNull . . . 360
isPositiveNumberOrNaOrInfVector . . . 361
isPositiveNumberOrNaOrInfVectorOrNull . . . 363
isPositiveNumberOrNaOrNanOrInfScalar . . . 364
isPositiveNumberOrNaOrNanOrInfScalarOrNull . . . 365
isPositiveNumberOrNaOrNanOrInfVector . . . 367
isPositiveNumberOrNaOrNanOrInfVectorOrNull . . . 368
isPositiveNumberOrNaOrNanScalar . . . 369
isPositiveNumberOrNaOrNanScalarOrNull . . . 370
isPositiveNumberOrNaOrNanVector . . . 372
isPositiveNumberOrNaOrNanVectorOrNull . . . 373
isPositiveNumberOrNaScalar . . . 374
isPositiveNumberOrNaScalarOrNull . . . 376
isPositiveNumberOrNaVector . . . 377
isPositiveNumberOrNaVectorOrNull . . . 378
isPositiveNumberScalar . . . 380
isPositiveNumberScalarOrNull . . . 381
isPositiveNumberVector . . . 382
isPositiveNumberVectorOrNull . . . 384
isStrictlyNegativeIntegerOrInfScalar . . . 385
isStrictlyNegativeIntegerOrInfScalarOrNull . . . 386
isStrictlyNegativeIntegerOrInfVector . . . 387
isStrictlyNegativeIntegerOrInfVectorOrNull . . . 389
isStrictlyNegativeIntegerOrNanOrInfScalar . . . 390
isStrictlyNegativeIntegerOrNanOrInfScalarOrNull . . . 391
isStrictlyNegativeIntegerOrNanOrInfVector . . . 393
isStrictlyNegativeIntegerOrNanOrInfVectorOrNull . . . 394
isStrictlyNegativeIntegerOrNanScalar . . . 395
isStrictlyNegativeIntegerOrNanScalarOrNull . . . 396

isStrictlyNegativeIntegerOrNanVector . . . 398
isStrictlyNegativeIntegerOrNanVectorOrNull . . . 399
isStrictlyNegativeIntegerOrNaOrInfScalar . . . 400
isStrictlyNegativeIntegerOrNaOrInfScalarOrNull . . . 402
isStrictlyNegativeIntegerOrNaOrInfVector . . . 403
isStrictlyNegativeIntegerOrNaOrInfVectorOrNull . . . 404
isStrictlyNegativeIntegerOrNaOrNanOrInfScalar . . . 406
isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull . . . 407
isStrictlyNegativeIntegerOrNaOrNanOrInfVector . . . 408
isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull . . . 410
isStrictlyNegativeIntegerOrNaOrNanScalar . . . 411
isStrictlyNegativeIntegerOrNaOrNanScalarOrNull . . . 412
isStrictlyNegativeIntegerOrNaOrNanVector . . . 413
isStrictlyNegativeIntegerOrNaOrNanVectorOrNull . . . 415
isStrictlyNegativeIntegerOrNaScalar . . . 416
isStrictlyNegativeIntegerOrNaScalarOrNull . . . 417
isStrictlyNegativeIntegerOrNaVector . . . 419
isStrictlyNegativeIntegerOrNaVectorOrNull . . . 420
isStrictlyNegativeIntegerScalar . . . 421
isStrictlyNegativeIntegerScalarOrNull . . . 422
isStrictlyNegativeIntegerVector . . . 424
isStrictlyNegativeIntegerVectorOrNull . . . 425
isStrictlyNegativeNumberOrInfScalar . . . 426
isStrictlyNegativeNumberOrInfScalarOrNull . . . 428
isStrictlyNegativeNumberOrInfVector . . . 429
isStrictlyNegativeNumberOrInfVectorOrNull . . . 430
isStrictlyNegativeNumberOrNanOrInfScalar . . . 432
isStrictlyNegativeNumberOrNanOrInfScalarOrNull . . . 433
isStrictlyNegativeNumberOrNanOrInfVector . . . 434
isStrictlyNegativeNumberOrNanOrInfVectorOrNull . . . 436
isStrictlyNegativeNumberOrNanScalar . . . 437
isStrictlyNegativeNumberOrNanScalarOrNull . . . 438
isStrictlyNegativeNumberOrNanVector . . . 439
isStrictlyNegativeNumberOrNanVectorOrNull . . . 441
isStrictlyNegativeNumberOrNaOrInfScalar . . . 442
isStrictlyNegativeNumberOrNaOrInfScalarOrNull . . . 443
isStrictlyNegativeNumberOrNaOrInfVector . . . 445
isStrictlyNegativeNumberOrNaOrInfVectorOrNull . . . 446
isStrictlyNegativeNumberOrNaOrNanOrInfScalar . . . 447
isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull . . . 448
isStrictlyNegativeNumberOrNaOrNanOrInfVector . . . 450
isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull . . . 451
isStrictlyNegativeNumberOrNaOrNanScalar . . . 452
isStrictlyNegativeNumberOrNaOrNanScalarOrNull . . . 454
isStrictlyNegativeNumberOrNaOrNanVector . . . 455
isStrictlyNegativeNumberOrNaOrNanVectorOrNull . . . 456
isStrictlyNegativeNumberOrNaScalar . . . 458
isStrictlyNegativeNumberOrNaScalarOrNull . . . 459

isStrictlyNegativeNumberOrNaVector . . . 460
isStrictlyNegativeNumberOrNaVectorOrNull . . . 462
isStrictlyNegativeNumberScalar . . . 463
isStrictlyNegativeNumberScalarOrNull . . . 464
isStrictlyNegativeNumberVector . . . 465
isStrictlyNegativeNumberVectorOrNull . . . 467
isStrictlyPositiveIntegerOrInfScalar . . . 468
isStrictlyPositiveIntegerOrInfScalarOrNull . . . 469
isStrictlyPositiveIntegerOrInfVector . . . 471
isStrictlyPositiveIntegerOrInfVectorOrNull . . . 472
isStrictlyPositiveIntegerOrNanOrInfScalar . . . 473
isStrictlyPositiveIntegerOrNanOrInfScalarOrNull . . . 474
isStrictlyPositiveIntegerOrNanOrInfVector . . . 476
isStrictlyPositiveIntegerOrNanOrInfVectorOrNull . . . 477
isStrictlyPositiveIntegerOrNanScalar . . . 478
isStrictlyPositiveIntegerOrNanScalarOrNull . . . 480
isStrictlyPositiveIntegerOrNanVector . . . 481
isStrictlyPositiveIntegerOrNanVectorOrNull . . . 482
isStrictlyPositiveIntegerOrNaOrInfScalar . . . 484
isStrictlyPositiveIntegerOrNaOrInfScalarOrNull . . . 485
isStrictlyPositiveIntegerOrNaOrInfVector . . . 486
isStrictlyPositiveIntegerOrNaOrInfVectorOrNull . . . 488
isStrictlyPositiveIntegerOrNaOrNanOrInfScalar . . . 489
isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull . . . 490
isStrictlyPositiveIntegerOrNaOrNanOrInfVector . . . 491
isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull . . . 493
isStrictlyPositiveIntegerOrNaOrNanScalar . . . 494
isStrictlyPositiveIntegerOrNaOrNanScalarOrNull . . . 495
isStrictlyPositiveIntegerOrNaOrNanVector . . . 497
isStrictlyPositiveIntegerOrNaOrNanVectorOrNull . . . 498
isStrictlyPositiveIntegerOrNaScalar . . . 499
isStrictlyPositiveIntegerOrNaScalarOrNull . . . 500
isStrictlyPositiveIntegerOrNaVector . . . 502
isStrictlyPositiveIntegerOrNaVectorOrNull . . . 503
isStrictlyPositiveIntegerScalar . . . 504
isStrictlyPositiveIntegerScalarOrNull . . . 506
isStrictlyPositiveIntegerVector . . . 507
isStrictlyPositiveIntegerVectorOrNull . . . 508
isStrictlyPositiveNumberOrInfScalar . . . 510
isStrictlyPositiveNumberOrInfScalarOrNull . . . 511
isStrictlyPositiveNumberOrInfVector . . . 512
isStrictlyPositiveNumberOrInfVectorOrNull . . . 514
isStrictlyPositiveNumberOrNanOrInfScalar . . . 515
isStrictlyPositiveNumberOrNanOrInfScalarOrNull . . . 516
isStrictlyPositiveNumberOrNanOrInfVector . . . 517
isStrictlyPositiveNumberOrNanOrInfVectorOrNull . . . 519
isStrictlyPositiveNumberOrNanScalar . . . 520
isStrictlyPositiveNumberOrNanScalarOrNull . . . 521

isStrictlyPositiveNumberOrNanVector . . . 523
isStrictlyPositiveNumberOrNanVectorOrNull . . . 524
isStrictlyPositiveNumberOrNaOrInfScalar . . . 525
isStrictlyPositiveNumberOrNaOrInfScalarOrNull . . . 526
isStrictlyPositiveNumberOrNaOrInfVector . . . 528
isStrictlyPositiveNumberOrNaOrInfVectorOrNull . . . 529
isStrictlyPositiveNumberOrNaOrNanOrInfScalar . . . 530
isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull . . . 532
isStrictlyPositiveNumberOrNaOrNanOrInfVector . . . 533
isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull . . . 534
isStrictlyPositiveNumberOrNaOrNanScalar . . . 536
isStrictlyPositiveNumberOrNaOrNanScalarOrNull . . . 537
isStrictlyPositiveNumberOrNaOrNanVector . . . 538
isStrictlyPositiveNumberOrNaOrNanVectorOrNull . . . 540
isStrictlyPositiveNumberOrNaScalar . . . 541
isStrictlyPositiveNumberOrNaScalarOrNull . . . 542
isStrictlyPositiveNumberOrNaVector . . . 543
isStrictlyPositiveNumberOrNaVectorOrNull . . . 545
isStrictlyPositiveNumberScalar . . . 546
isStrictlyPositiveNumberScalarOrNull . . . 547
isStrictlyPositiveNumberVector . . . 549
isStrictlyPositiveNumberVectorOrNull . . . 550
isStringScalar . . . 551
isStringScalarOrNull . . . 552
isStringVector . . . 554
isStringVectorOrNull . . . 555
isZeroOrNanScalar . . . 556
isZeroOrNanScalarOrNull . . . 557
isZeroOrNanVector . . . 559
isZeroOrNanVectorOrNull . . . 560
isZeroOrNaOrNanScalar . . . 561
isZeroOrNaOrNanScalarOrNull . . . 562
isZeroOrNaOrNanVector . . . 564
isZeroOrNaOrNanVectorOrNull . . . 565
isZeroOrNaScalar . . . 566
isZeroOrNaScalarOrNull . . . 567
isZeroOrNaVector . . . 569
isZeroOrNaVectorOrNull . . . 570
isZeroScalar . . . 571
isZeroScalarOrNull . . . 572
isZeroVector . . . 573
isZeroVectorOrNull . . . 575

Index **577**

checkarg *Perform a basic check on the type of an argument and its value(s) and set a default value if applicable.*

Description

This function is the main function that all isXxx functions are wrappers around, each with specific parameter settings. It can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
checkarg(argument, content, default = NULL, stopIfNot = FALSE,
  nullAllowed = FALSE, n = 1, naAllowed = FALSE, nanAllowed = TRUE,
  infAllowed = TRUE, nonIntegerAllowed = TRUE, negativeAllowed = TRUE,
  zeroAllowed = TRUE, positiveAllowed = TRUE, emptyStringAllowed = TRUE,
  message = "Argument \\1has invalid value", argumentName = "")
```

Arguments

`argument`	Argument to check.
`content`	Required content of argument, case-insensitive. Either "BOOLEAN" (short: "B") for an argument of primitive type logical, "NUMBER" (short: "N") for an argument of primitive type numeric, or "STRING" (short: "S") for an argument of primitive type character.
`default`	If not NULL and the argument is NULL, the default value is returned instead of the result of the content checks (TRUE or FALSE). Parameter stopIfNot must be set to TRUE in this case. The content of the default value is not checked.
`stopIfNot`	If TRUE and the argument checks fail, an exception is be thrown. Must be set to TRUE if parameter default is not NULL.
`nullAllowed`	If TRUE, argument may be NULL.
`n`	Number of elements the argument must have. Default: n = 1, i.e. a scalar value is expected. If NA, any length is allowed. If the argument has zero elements and either n = 0 or n = NA, only the primitive type is checked and no other checks on content are applied.

`naAllowed` If TRUE, NA value(s) are allowed. If FALSE, NaN value(s), which also test true for is.na, are ignored and can be further constrained by nanAllowed.

`nanAllowed` If TRUE, NaN value(s) are allowed.

`infAllowed` If TRUE, Inf value(s) are allowed.

`nonIntegerAllowed` If TRUE, non-integer value(s) are allowed. If FALSE, NA, NaN and Inf value(s), which are not integers, are ignored and can instead be constrained further by respectively naAllowed, nanAllowed and infAllowed.

`negativeAllowed` If TRUE, negative value(s) are allowed.

`zeroAllowed` If TRUE, zero value(s) are allowed.

`positiveAllowed` If TRUE, positive value(s) are allowed.

`emptyStringAllowed` If TRUE, empty string value(s) are allowed.

`message` The message provided when an exception is thrown. The first instance of \1 is replaced with '\'argumentName\' ', if the latter is not empty. If NULL, the same default message is used.

`argumentName` The name of the variable to be used in the exception message. If NULL, the same default argumentName is used.

Value

If no default is provided, i.e. if the default parameter is not null): TRUE is returned if the argument passes the checks and otherwise FALSE. If a default is provided, the default is returned in case the argument is null and otherwise the argument is returned.

Examples

```
checkarg(TRUE, "BOOLEAN")
  # returns TRUE (argument is valid)
checkarg(FALSE, "BOOLEAN")
  # returns TRUE (argument is valid)
checkarg(1, "BOOLEAN")
  # returns FALSE (argument is invalid)
checkarg("Y", "BOOLEAN")
  # returns FALSE (argument is invalid)
#checkarg("Y", "BOOLEAN", stopIfNot = TRUE)
  # throws exception with message defined by message and argumentName parameters

checkarg(1, "NUMBER", default = 0)
  # returns 1 (argument is returned if provided, i.e. not NULL, and valid)
checkarg(NULL, "NUMBER", default = 0)
  # returns 0 (default is returned since argument is not provided)
checkarg(NULL, "NUMBER", default = 0)
  # returns 0 (default is returned since argument is not provided)
checkarg(NA, "NUMBER")
  # returns FALSE (NA values are not allowed by default)
```

```
checkarg(NA, "NUMBER", naAllowed = TRUE)
  # returns TRUE (NA values are allowed)
checkarg(c(0, 1), "NUMBER")
  # returns FALSE (scalar value expected by default)
checkarg(c(0, 1), "NUMBER", n = NA)
  # returns TRUE (any length vector allowed)

checkarg("X", "STRING")
  # returns TRUE (argument is valid)
checkarg(TRUE, "STRING")
  # returns FALSE (argument is invalid)
checkarg(1, "STRING")
  # returns FALSE (argument is invalid)
```

`checkargFailedHandler` *Helper function for the checkarg function, called in cases the argument does not pass the check. Throws an exception if stopIfNot is TRUE.*

Description

Helper function for the checkarg function, called in cases the argument does not pass the check. Throws an exception if stopIfNot is TRUE.

Usage

```
checkargFailedHandler(default, stopIfNot, messagePattern, argumentName)
```

Arguments

`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`messagePattern`	See checkarg function.
`argumentName`	See checkarg function.

Value

FALSE.

`isBooleanOrNaScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, naAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanOrNaScalar(FALSE)
   # returns TRUE (argument is valid)
isBooleanOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isBooleanOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaScalar(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanOrNaScalar("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaScalar(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

---

isBooleanOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanOrNaScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, naAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanOrNaScalarOrNull(FALSE)
   # returns TRUE (argument is valid)
isBooleanOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isBooleanOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaScalarOrNull(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanOrNaScalarOrNull("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaScalarOrNull(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

---

isBooleanOrNaVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanOrNaVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, naAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanOrNaVector(FALSE)
   # returns TRUE (argument is valid)
isBooleanOrNaVector("X")
   # returns FALSE (argument is invalid)
#isBooleanOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaVector(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanOrNaVector("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaVector(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

---

isBooleanOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanOrNaVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, naAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanOrNaVectorOrNull(FALSE)
   # returns TRUE (argument is valid)
isBooleanOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isBooleanOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaVectorOrNull(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanOrNaVectorOrNull("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanOrNaVectorOrNull(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

`isBooleanScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, naAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanScalar(FALSE)
   # returns TRUE (argument is valid)
isBooleanScalar("X")
   # returns FALSE (argument is invalid)
#isBooleanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanScalar(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanScalar("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanScalar(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

---

isBooleanScalarOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, naAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanScalarOrNull(FALSE)
   # returns TRUE (argument is valid)
isBooleanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isBooleanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanScalarOrNull(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanScalarOrNull("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanScalarOrNull(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

---

isBooleanVector — *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, naAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanVector(FALSE)
   # returns TRUE (argument is valid)
isBooleanVector("X")
   # returns FALSE (argument is invalid)
#isBooleanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanVector(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanVector("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanVector(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

`isBooleanVectorOrNull` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isBooleanVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "B", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, naAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isBooleanVectorOrNull(FALSE)
   # returns TRUE (argument is valid)
isBooleanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isBooleanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanVectorOrNull(FALSE, default = TRUE)
   # returns FALSE (the argument, rather than the default, since it is not NULL)
#isBooleanVectorOrNull("X", default = TRUE)
   # throws exception with message defined by message and argumentName parameters
isBooleanVectorOrNull(NULL, default = TRUE)
   # returns TRUE (the default, rather than the argument, since it is NULL)
```

`isIntegerOrInfScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrInfScalar(2)
   # returns TRUE (argument is valid)
isIntegerOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrInfScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrInfVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrInfVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrInfVector(2)
   # returns TRUE (argument is valid)
isIntegerOrInfVector("X")
   # returns FALSE (argument is invalid)
#isIntegerOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrInfVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerOrNanOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isIntegerOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanOrInfScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isIntegerOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerOrNanOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanOrInfVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNanScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanScalar(2)
   # returns TRUE (argument is valid)
isIntegerOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNanVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanVector(2)
   # returns TRUE (argument is valid)
isIntegerOrNanVector("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isIntegerOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerOrNaOrInfScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrInfScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerOrNaOrInfVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrInfVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerOrNaOrNanOrInfVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerOrNaOrNanScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaOrNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerOrNaScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaScalar(2)
   # returns TRUE (argument is valid)
isIntegerOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaVector(2)
   # returns TRUE (argument is valid)
isIntegerOrNaVector("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerOrNaVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isIntegerScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerScalar(2)
   # returns TRUE (argument is valid)
isIntegerScalar("X")
   # returns FALSE (argument is invalid)
#isIntegerScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerScalarOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerScalarOrNull(2)
   # returns TRUE (argument is valid)
isIntegerScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerVector — *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerVector(2)
   # returns TRUE (argument is valid)
isIntegerVector("X")
   # returns FALSE (argument is invalid)
#isIntegerVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isIntegerVectorOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isIntegerVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isIntegerVectorOrNull(2)
   # returns TRUE (argument is valid)
isIntegerVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isIntegerVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isIntegerVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isIntegerVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isIntegerVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNanScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNanScalar(argument, default = NULL, stopIfNot = FALSE, message = NULL,
  argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNanScalar(NaN)
   # returns TRUE (argument is valid)
isNanScalar("X")
   # returns FALSE (argument is invalid)
#isNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

---

isNanScalarOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNanScalarOrNull(NaN)
   # returns TRUE (argument is valid)
isNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

---

isNanVector — *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNanVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNanVector(NaN)
   # returns TRUE (argument is valid)
isNanVector("X")
   # returns FALSE (argument is invalid)
#isNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

---

isNanVectorOrNull — *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNanVectorOrNull(NaN)
   # returns TRUE (argument is valid)
isNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

`isNaOrNanScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNanScalar(NA)
   # returns TRUE (argument is valid)
isNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanScalar(NA, default = NaN)
   # returns NA (the argument, rather than the default, since it is not NULL)
#isNaOrNanScalar("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanScalar(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

---

isNaOrNanScalarOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNanScalarOrNull(NA)
   # returns TRUE (argument is valid)
isNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanScalarOrNull(NA, default = NaN)
```

```
   # returns NA (the argument, rather than the default, since it is not NULL)
#isNaOrNanScalarOrNull("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanScalarOrNull(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

`isNaOrNanVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNanVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNanVector(NA)
   # returns TRUE (argument is valid)
isNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanVector(NA, default = NaN)
   # returns NA (the argument, rather than the default, since it is not NULL)
#isNaOrNanVector("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanVector(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

---

isNaOrNanVectorOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNanVectorOrNull(NA)
   # returns TRUE (argument is valid)
isNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanVectorOrNull(NA, default = NaN)
   # returns NA (the argument, rather than the default, since it is not NULL)
#isNaOrNanVectorOrNull("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isNaOrNanVectorOrNull(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

---

isNaOrNonEmptyStringScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNonEmptyStringScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, naAllowed = TRUE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNonEmptyStringScalar("X")
   # returns TRUE (argument is valid)
isNaOrNonEmptyStringScalar(1)
   # returns FALSE (argument is invalid)
#isNaOrNonEmptyStringScalar(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringScalar("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrNonEmptyStringScalar(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringScalar(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

`isNaOrNonEmptyStringScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNonEmptyStringScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, naAllowed = TRUE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNonEmptyStringScalarOrNull("X")
   # returns TRUE (argument is valid)
isNaOrNonEmptyStringScalarOrNull(1)
   # returns FALSE (argument is invalid)
#isNaOrNonEmptyStringScalarOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringScalarOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrNonEmptyStringScalarOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringScalarOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNaOrNonEmptyStringVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNonEmptyStringVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, naAllowed = TRUE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNonEmptyStringVector("X")
   # returns TRUE (argument is valid)
isNaOrNonEmptyStringVector(1)
   # returns FALSE (argument is invalid)
#isNaOrNonEmptyStringVector(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringVector("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrNonEmptyStringVector(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringVector(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNaOrNonEmptyStringVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrNonEmptyStringVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, naAllowed = TRUE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrNonEmptyStringVectorOrNull("X")
   # returns TRUE (argument is valid)
isNaOrNonEmptyStringVectorOrNull(1)
   # returns FALSE (argument is invalid)
#isNaOrNonEmptyStringVectorOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringVectorOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrNonEmptyStringVectorOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrNonEmptyStringVectorOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNaOrStringScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrStringScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, naAllowed = TRUE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrStringScalar("X")
   # returns TRUE (argument is valid)
isNaOrStringScalar(1)
   # returns FALSE (argument is invalid)
#isNaOrStringScalar(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrStringScalar("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrStringScalar(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrStringScalar(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

`isNaOrStringScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrStringScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, naAllowed = TRUE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrStringScalarOrNull("X")
   # returns TRUE (argument is valid)
isNaOrStringScalarOrNull(1)
   # returns FALSE (argument is invalid)
#isNaOrStringScalarOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrStringScalarOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrStringScalarOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrStringScalarOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNaOrStringVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrStringVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, naAllowed = TRUE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrStringVector("X")
   # returns TRUE (argument is valid)
isNaOrStringVector(1)
   # returns FALSE (argument is invalid)
#isNaOrStringVector(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrStringVector("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrStringVector(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrStringVector(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNaOrStringVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaOrStringVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, naAllowed = TRUE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaOrStringVectorOrNull("X")
   # returns TRUE (argument is valid)
isNaOrStringVectorOrNull(1)
   # returns FALSE (argument is invalid)
#isNaOrStringVectorOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNaOrStringVectorOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNaOrStringVectorOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNaOrStringVectorOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNaScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaScalar(argument, default = NULL, stopIfNot = FALSE, message = NULL,
  argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaScalar(NA)
   # returns TRUE (argument is valid)
isNaScalar("X")
   # returns FALSE (argument is invalid)
#isNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

`isNaScalarOrNull` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaScalarOrNull(NA)
   # returns TRUE (argument is valid)
isNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

`isNaVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaVector(NA)
   # returns TRUE (argument is valid)
isNaVector("X")
   # returns FALSE (argument is invalid)
#isNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

---

isNaVectorOrNull — *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNaVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNaVectorOrNull(NA)
   # returns TRUE (argument is valid)
isNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

---

isNegativeIntegerOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrInfScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfScalarOrNull(-2, default = -1)
```

```
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrNanOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrNanScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrNaOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrNaOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isNegativeIntegerOrNaOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrNaOrNanScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrNaOrNanVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaScalarOrNull(-2, default = -1)
```

```
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerOrNaVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerOrNaVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerOrNaVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerOrNaVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerScalar(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeIntegerScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerVector(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerVector("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeIntegerVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeIntegerVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeIntegerVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeIntegerVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeIntegerVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeIntegerVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeIntegerVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isNegativeNumberOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberOrNanVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfScalarOrNull(-2, default = -1)
```

```
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberOrNaOrNanOrInfScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberOrNaOrNanScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberOrNaOrNanVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberOrNaVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isNegativeNumberOrNaVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberOrNaVectorOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberOrNaVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberOrNaVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberOrNaVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberScalar(-2)
   # returns TRUE (argument is valid)
isNegativeNumberScalar("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberScalarOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isNegativeNumberVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberVector(-2)
   # returns TRUE (argument is valid)
isNegativeNumberVector("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNegativeNumberVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNegativeNumberVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNegativeNumberVectorOrNull(-2)
   # returns TRUE (argument is valid)
isNegativeNumberVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNegativeNumberVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isNegativeNumberVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isNegativeNumberVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isNonEmptyStringScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonEmptyStringScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, naAllowed = FALSE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonEmptyStringScalar("X")
   # returns TRUE (argument is valid)
isNonEmptyStringScalar(1)
   # returns FALSE (argument is invalid)
#isNonEmptyStringScalar(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringScalar("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNonEmptyStringScalar(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringScalar(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNonEmptyStringScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonEmptyStringScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, naAllowed = FALSE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonEmptyStringScalarOrNull("X")
   # returns TRUE (argument is valid)
isNonEmptyStringScalarOrNull(1)
   # returns FALSE (argument is invalid)
#isNonEmptyStringScalarOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringScalarOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
```

```
#isNonEmptyStringScalarOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringScalarOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

`isNonEmptyStringVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonEmptyStringVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, naAllowed = FALSE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonEmptyStringVector("X")
   # returns TRUE (argument is valid)
isNonEmptyStringVector(1)
   # returns FALSE (argument is invalid)
#isNonEmptyStringVector(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringVector("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNonEmptyStringVector(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringVector(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNonEmptyStringVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonEmptyStringVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, naAllowed = FALSE, emptyStringAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonEmptyStringVectorOrNull("X")
   # returns TRUE (argument is valid)
isNonEmptyStringVectorOrNull(1)
   # returns FALSE (argument is invalid)
#isNonEmptyStringVectorOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringVectorOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isNonEmptyStringVectorOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isNonEmptyStringVectorOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrInfScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNanOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNanOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isNonZeroIntegerOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNaOrInfScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNaOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNaOrNanOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNaOrNanScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNaScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerOrNaVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerOrNaVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerScalar(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroIntegerVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerVector(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isNonZeroIntegerVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroIntegerVectorOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroIntegerVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroIntegerVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroIntegerVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroIntegerVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroIntegerVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroIntegerVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrInfScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrNanOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrNanScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrNaOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrNaOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isNonZeroNumberOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrNaOrNanScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrNaOrNanVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberOrNaVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberOrNaVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberScalar(2)
   # returns TRUE (argument is valid)
isNonZeroNumberScalar("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNonZeroNumberScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberScalarOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberVector(2)
   # returns TRUE (argument is valid)
isNonZeroNumberVector("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNonZeroNumberVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNonZeroNumberVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNonZeroNumberVectorOrNull(2)
   # returns TRUE (argument is valid)
isNonZeroNumberVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNonZeroNumberVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNonZeroNumberVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNonZeroNumberVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrInfScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrInfScalar(2)
   # returns TRUE (argument is valid)
isNumberOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNumberOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrInfScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrInfVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrInfVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrInfVector(2)
   # returns TRUE (argument is valid)
isNumberOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNumberOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrInfVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrNanOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isNumberOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanOrInfScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isNumberOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrNanOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanOrInfVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNanScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanScalar(2)
   # returns TRUE (argument is valid)
isNumberOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNanVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanVector(2)
   # returns TRUE (argument is valid)
isNumberOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isNumberOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrNaOrInfScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrInfScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrNaOrInfVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrInfVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrNaOrNanOrInfVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrNaOrNanScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaOrNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberOrNaScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaScalar(2)
   # returns TRUE (argument is valid)
isNumberOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaVector *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaVector(2)
   # returns TRUE (argument is valid)
isNumberOrNaVector("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberOrNaVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isNumberScalar` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberScalar(2)
   # returns TRUE (argument is valid)
isNumberScalar("X")
   # returns FALSE (argument is invalid)
#isNumberScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberScalarOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberScalarOrNull(2)
   # returns TRUE (argument is valid)
isNumberScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberVector — *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberVector(2)
   # returns TRUE (argument is valid)
isNumberVector("X")
   # returns FALSE (argument is invalid)
#isNumberVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isNumberVectorOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isNumberVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = TRUE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isNumberVectorOrNull(2)
   # returns TRUE (argument is valid)
isNumberVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isNumberVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isNumberVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isNumberVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isNumberVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isPositiveIntegerOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrNanScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrNanVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrNaOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrNaOrNanOrInfScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrNaOrNanScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerOrNaOrNanVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isPositiveIntegerOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerScalar(2)
   # returns TRUE (argument is valid)
isPositiveIntegerScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveIntegerVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerVector(2)
   # returns TRUE (argument is valid)
isPositiveIntegerVector("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveIntegerVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveIntegerVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveIntegerVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveIntegerVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveIntegerVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveIntegerVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveIntegerVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrInfVector`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrNanOrInfScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrNanScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrNanVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrInfScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrInfVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isPositiveNumberOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrNaOrNanOrInfScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrNaOrNanOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrNaOrNanVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberOrNaScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberScalar(2)
   # returns TRUE (argument is valid)
isPositiveNumberScalar("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberScalarOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isPositiveNumberVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberVector(2)
   # returns TRUE (argument is valid)
isPositiveNumberVector("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isPositiveNumberVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isPositiveNumberVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isPositiveNumberVectorOrNull(2)
   # returns TRUE (argument is valid)
isPositiveNumberVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isPositiveNumberVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isPositiveNumberVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isPositiveNumberVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isStrictlyNegativeIntegerOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeIntegerOrNanOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanScalarOrNull(-2, default = -1)
```

```
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeIntegerOrNanVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeIntegerOrNaOrInfScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeIntegerOrNaOrNanOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isStrictlyNegativeIntegerOrNaOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeIntegerOrNaVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerOrNaVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerOrNaVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerOrNaVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerScalarOrNull(-2, default = -1)
```

```
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeIntegerVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeIntegerVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeIntegerVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeIntegerVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeIntegerVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeIntegerVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeIntegerVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeIntegerVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeNumberOrInfScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeNumberOrNanOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeNumberOrNanOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isStrictlyNegativeNumberOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeNumberOrNaOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanOrInfScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanOrInfScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull(-2, default = -1)
```

```
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanOrInfVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanOrInfVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeNumberOrNaOrNanScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaOrNanVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaOrNanVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaOrNanVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeNumberOrNaScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberOrNaVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyNegativeNumberOrNaVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberOrNaVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberOrNaVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberOrNaVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberScalar(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberScalar(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberScalar("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberScalar(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberScalarOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberScalarOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberScalarOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberScalarOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberVector(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isStrictlyNegativeNumberVector(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberVector("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberVector(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyNegativeNumberVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyNegativeNumberVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = TRUE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyNegativeNumberVectorOrNull(-2)
   # returns TRUE (argument is valid)
isStrictlyNegativeNumberVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyNegativeNumberVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberVectorOrNull(-2, default = -1)
   # returns -2 (the argument, rather than the default, since it is not NULL)
#isStrictlyNegativeNumberVectorOrNull("X", default = -1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyNegativeNumberVectorOrNull(NULL, default = -1)
   # returns -1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerOrInfScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerOrNanOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerOrNanScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerOrNaOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerOrNaOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isStrictlyPositiveIntegerOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

isStrictlyPositiveIntegerOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerOrNaVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveIntegerScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveIntegerVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveIntegerVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = FALSE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveIntegerVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveIntegerVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveIntegerVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveIntegerVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveIntegerVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberOrInfScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberOrInfVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNanOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isStrictlyPositiveNumberOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNanScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberOrNanVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrInfScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfScalarOrNull(2, default = 1)
```

```
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberOrNaOrInfVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrNanOrInfScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanOrInfScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanOrInfScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanOrInfScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanOrInfScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanOrInfScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrNanOrInfVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanOrInfVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanOrInfVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanOrInfVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanOrInfVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanOrInfVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberOrNaOrNanScalar`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaOrNanVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberOrNaOrNanVectorOrNull`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaOrNanVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaOrNanVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaOrNanVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaOrNanVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaScalar(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaVector

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaVector(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

```
isStrictlyPositiveNumberOrNaVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberOrNaVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberOrNaVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberOrNaVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberOrNaVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberScalar

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberScalar(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberScalar("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberScalar(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberScalar("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberScalar(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberScalarOrNull(argument, default = NULL,
  stopIfNot = FALSE, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberScalarOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberScalarOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberScalarOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberScalarOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

`isStrictlyPositiveNumberVector`
Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberVector(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberVector(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberVector("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberVector(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberVector("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberVector(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStrictlyPositiveNumberVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStrictlyPositiveNumberVectorOrNull(argument, default = NULL,
  stopIfNot = FALSE, n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = FALSE, negativeAllowed = FALSE, positiveAllowed = TRUE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStrictlyPositiveNumberVectorOrNull(2)
   # returns TRUE (argument is valid)
isStrictlyPositiveNumberVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isStrictlyPositiveNumberVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberVectorOrNull(2, default = 1)
   # returns 2 (the argument, rather than the default, since it is not NULL)
#isStrictlyPositiveNumberVectorOrNull("X", default = 1)
   # throws exception with message defined by message and argumentName parameters
isStrictlyPositiveNumberVectorOrNull(NULL, default = 1)
   # returns 1 (the default, rather than the argument, since it is NULL)
```

---

isStringScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStringScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, naAllowed = FALSE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStringScalar("X")
   # returns TRUE (argument is valid)
isStringScalar(1)
   # returns FALSE (argument is invalid)
#isStringScalar(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStringScalar("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isStringScalar(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isStringScalar(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isStringScalarOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStringScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, naAllowed = FALSE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStringScalarOrNull("X")
   # returns TRUE (argument is valid)
isStringScalarOrNull(1)
   # returns FALSE (argument is invalid)
#isStringScalarOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStringScalarOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isStringScalarOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isStringScalarOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

`isStringVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStringVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, naAllowed = FALSE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStringVector("X")
   # returns TRUE (argument is valid)
isStringVector(1)
   # returns FALSE (argument is invalid)
#isStringVector(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStringVector("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isStringVector(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isStringVector(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isStringVectorOrNull *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isStringVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "S", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, naAllowed = FALSE, emptyStringAllowed = TRUE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isStringVectorOrNull("X")
   # returns TRUE (argument is valid)
isStringVectorOrNull(1)
   # returns FALSE (argument is invalid)
#isStringVectorOrNull(1, stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isStringVectorOrNull("X", default = "Y")
   # returns "X" (the argument, rather than the default, since it is not NULL)
#isStringVectorOrNull(1, default = "Y")
   # throws exception with message defined by message and argumentName parameters
isStringVectorOrNull(NULL, default = "Y")
   # returns "Y" (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNanScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNanScalar(0)
   # returns TRUE (argument is valid)
isZeroOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isZeroOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanScalar(0, default = NaN)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNanScalar("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanScalar(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

`isZeroOrNanScalarOrNull`

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNanScalarOrNull(0)
   # returns TRUE (argument is valid)
isZeroOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanScalarOrNull(0, default = NaN)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNanScalarOrNull("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanScalarOrNull(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

`isZeroOrNanVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNanVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNanVector(0)
   # returns TRUE (argument is valid)
isZeroOrNanVector("X")
   # returns FALSE (argument is invalid)
#isZeroOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanVector(0, default = NaN)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNanVector("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanVector(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNanVectorOrNull(0)
   # returns TRUE (argument is valid)
isZeroOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanVectorOrNull(0, default = NaN)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNanVectorOrNull("X", default = NaN)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNanVectorOrNull(NULL, default = NaN)
   # returns NaN (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNaOrNanScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaOrNanScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaOrNanScalar(0)
   # returns TRUE (argument is valid)
isZeroOrNaOrNanScalar("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaOrNanScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanScalar(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaOrNanScalar("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanScalar(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNaOrNanScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaOrNanScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaOrNanScalarOrNull(0)
   # returns TRUE (argument is valid)
isZeroOrNaOrNanScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaOrNanScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanScalarOrNull(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaOrNanScalarOrNull("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanScalarOrNull(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

`isZeroOrNaOrNanVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaOrNanVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaOrNanVector(0)
   # returns TRUE (argument is valid)
isZeroOrNaOrNanVector("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaOrNanVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanVector(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaOrNanVector("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanVector(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNaOrNanVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaOrNanVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = TRUE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaOrNanVectorOrNull(0)
   # returns TRUE (argument is valid)
isZeroOrNaOrNanVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaOrNanVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanVectorOrNull(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaOrNanVectorOrNull("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaOrNanVectorOrNull(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNaScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaScalar(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaScalar(0)
   # returns TRUE (argument is valid)
isZeroOrNaScalar("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaScalar(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaScalar("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaScalar(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNaScalarOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaScalarOrNull(0)
   # returns TRUE (argument is valid)
isZeroOrNaScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaScalarOrNull(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaScalarOrNull("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaScalarOrNull(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

`isZeroOrNaVector` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaVector(0)
   # returns TRUE (argument is valid)
isZeroOrNaVector("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaVector(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaVector("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaVector(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

---

isZeroOrNaVectorOrNull

Wrapper for the checkarg function, using specific parameter settings.

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroOrNaVectorOrNull(argument, default = NULL, stopIfNot = FALSE,
  n = NA, message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = TRUE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroOrNaVectorOrNull(0)
   # returns TRUE (argument is valid)
isZeroOrNaVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroOrNaVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaVectorOrNull(0, default = NA)
   # returns 0 (the argument, rather than the default, since it is not NULL)
#isZeroOrNaVectorOrNull("X", default = NA)
   # throws exception with message defined by message and argumentName parameters
isZeroOrNaVectorOrNull(NULL, default = NA)
   # returns NA (the default, rather than the argument, since it is NULL)
```

---

isZeroScalar *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroScalar(argument, default = NULL, stopIfNot = FALSE, message = NULL,
  argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroScalar(0)
   # returns TRUE (argument is valid)
isZeroScalar("X")
   # returns FALSE (argument is invalid)
#isZeroScalar("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

`isZeroScalarOrNull` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroScalarOrNull(argument, default = NULL, stopIfNot = FALSE,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = 1, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroScalarOrNull(0)
   # returns TRUE (argument is valid)
isZeroScalarOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroScalarOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

---

isZeroVector — *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroVector(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

argument	See checkarg function.
default	See checkarg function.
stopIfNot	See checkarg function.
n	See checkarg function.
message	See checkarg function.
argumentName	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = FALSE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroVector(0)
   # returns TRUE (argument is valid)
isZeroVector("X")
   # returns FALSE (argument is invalid)
#isZeroVector("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

`isZeroVectorOrNull` *Wrapper for the checkarg function, using specific parameter settings.*

Description

This function can be used in 3 ways:

1. Return TRUE or FALSE depending on whether the argument checks are passed. This is suitable e.g. for if statements that take further action if the argument does not pass the checks.

2. Throw an exception if the argument does not pass the checks. This is suitable e.g. when no further action needs to be taken other than throwing an exception if the argument does not pass the checks.

3. Same as (2) but by supplying a default value, a default can be assigned in a single statement, when the argument is NULL. The checks are still performed on the returned value, and an exception is thrown when not passed.

Usage

```
isZeroVectorOrNull(argument, default = NULL, stopIfNot = FALSE, n = NA,
  message = NULL, argumentName = NULL)
```

Arguments

`argument`	See checkarg function.
`default`	See checkarg function.
`stopIfNot`	See checkarg function.
`n`	See checkarg function.
`message`	See checkarg function.
`argumentName`	See checkarg function.

Details

Actual call to checkarg: checkarg(argument, "N", default = default, stopIfNot = stopIfNot, nullAllowed = TRUE, n = NA, zeroAllowed = TRUE, negativeAllowed = FALSE, positiveAllowed = FALSE, nonIntegerAllowed = TRUE, naAllowed = FALSE, nanAllowed = FALSE, infAllowed = FALSE, message = message, argumentName = argumentName)

Value

See checkarg function.

Examples

```
isZeroVectorOrNull(0)
   # returns TRUE (argument is valid)
isZeroVectorOrNull("X")
   # returns FALSE (argument is invalid)
#isZeroVectorOrNull("X", stopIfNot = TRUE)
   # throws exception with message defined by message and argumentName parameters
```

Index

checkarg, 11
checkargFailedHandler, 13

isBooleanOrNaScalar, 14
isBooleanOrNaScalarOrNull, 15
isBooleanOrNaVector, 16
isBooleanOrNaVectorOrNull, 17
isBooleanScalar, 19
isBooleanScalarOrNull, 20
isBooleanVector, 21
isBooleanVectorOrNull, 22
isIntegerOrInfScalar, 24
isIntegerOrInfScalarOrNull, 25
isIntegerOrInfVector, 26
isIntegerOrInfVectorOrNull, 27
isIntegerOrNanOrInfScalar, 29
isIntegerOrNanOrInfScalarOrNull, 30
isIntegerOrNanOrInfVector, 31
isIntegerOrNanOrInfVectorOrNull, 33
isIntegerOrNanScalar, 34
isIntegerOrNanScalarOrNull, 35
isIntegerOrNanVector, 36
isIntegerOrNanVectorOrNull, 38
isIntegerOrNaOrInfScalar, 39
isIntegerOrNaOrInfScalarOrNull, 40
isIntegerOrNaOrInfVector, 42
isIntegerOrNaOrInfVectorOrNull, 43
isIntegerOrNaOrNanOrInfScalar, 44
isIntegerOrNaOrNanOrInfScalarOrNull, 45
isIntegerOrNaOrNanOrInfVector, 47
isIntegerOrNaOrNanOrInfVectorOrNull, 48
isIntegerOrNaOrNanScalar, 49
isIntegerOrNaOrNanScalarOrNull, 51
isIntegerOrNaOrNanVector, 52
isIntegerOrNaOrNanVectorOrNull, 53
isIntegerOrNaScalar, 55
isIntegerOrNaScalarOrNull, 56
isIntegerOrNaVector, 57
isIntegerOrNaVectorOrNull, 58
isIntegerScalar, 60
isIntegerScalarOrNull, 61
isIntegerVector, 62
isIntegerVectorOrNull, 63
isNanScalar, 65
isNanScalarOrNull, 66
isNanVector, 67
isNanVectorOrNull, 68
isNaOrNanScalar, 69
isNaOrNanScalarOrNull, 70
isNaOrNanVector, 72
isNaOrNanVectorOrNull, 73
isNaOrNonEmptyStringScalar, 74
isNaOrNonEmptyStringScalarOrNull, 76
isNaOrNonEmptyStringVector, 77
isNaOrNonEmptyStringVectorOrNull, 78
isNaOrStringScalar, 79
isNaOrStringScalarOrNull, 81
isNaOrStringVector, 82
isNaOrStringVectorOrNull, 83
isNaScalar, 84
isNaScalarOrNull, 86
isNaVector, 87
isNaVectorOrNull, 88
isNegativeIntegerOrInfScalar, 89
isNegativeIntegerOrInfScalarOrNull, 90
isNegativeIntegerOrInfVector, 92
isNegativeIntegerOrInfVectorOrNull, 93
isNegativeIntegerOrNanOrInfScalar, 94
isNegativeIntegerOrNanOrInfScalarOrNull, 95
isNegativeIntegerOrNanOrInfVector, 97
isNegativeIntegerOrNanOrInfVectorOrNull, 98
isNegativeIntegerOrNanScalar, 99
isNegativeIntegerOrNanScalarOrNull, 101
isNegativeIntegerOrNanVector, 102

isNegativeIntegerOrNanVectorOrNull, 103
isNegativeIntegerOrNaOrInfScalar, 105
isNegativeIntegerOrNaOrInfScalarOrNull, 106
isNegativeIntegerOrNaOrInfVector, 107
isNegativeIntegerOrNaOrInfVectorOrNull, 109
isNegativeIntegerOrNaOrNanOrInfScalar, 110
isNegativeIntegerOrNaOrNanOrInfScalarOrNull, 111
isNegativeIntegerOrNaOrNanOrInfVector, 112
isNegativeIntegerOrNaOrNanOrInfVectorOrNull, 114
isNegativeIntegerOrNaOrNanScalar, 115
isNegativeIntegerOrNaOrNanScalarOrNull, 116
isNegativeIntegerOrNaOrNanVector, 118
isNegativeIntegerOrNaOrNanVectorOrNull, 119
isNegativeIntegerOrNaScalar, 120
isNegativeIntegerOrNaScalarOrNull, 121
isNegativeIntegerOrNaVector, 123
isNegativeIntegerOrNaVectorOrNull, 124
isNegativeIntegerScalar, 125
isNegativeIntegerScalarOrNull, 127
isNegativeIntegerVector, 128
isNegativeIntegerVectorOrNull, 129
isNegativeNumberOrInfScalar, 131
isNegativeNumberOrInfScalarOrNull, 132
isNegativeNumberOrInfVector, 133
isNegativeNumberOrInfVectorOrNull, 135
isNegativeNumberOrNanOrInfScalar, 136
isNegativeNumberOrNanOrInfScalarOrNull, 137
isNegativeNumberOrNanOrInfVector, 138
isNegativeNumberOrNanOrInfVectorOrNull, 140
isNegativeNumberOrNanScalar, 141
isNegativeNumberOrNanScalarOrNull, 142
isNegativeNumberOrNanVector, 144
isNegativeNumberOrNanVectorOrNull, 145
isNegativeNumberOrNaOrInfScalar, 146
isNegativeNumberOrNaOrInfScalarOrNull, 147
isNegativeNumberOrNaOrInfVector, 149
isNegativeNumberOrNaOrInfVectorOrNull, 150
isNegativeNumberOrNaOrNanOrInfScalar, 151
isNegativeNumberOrNaOrNanOrInfScalarOrNull, 153
isNegativeNumberOrNaOrNanOrInfVector, 154
isNegativeNumberOrNaOrNanOrInfVectorOrNull, 155
isNegativeNumberOrNaOrNanScalar, 157
isNegativeNumberOrNaOrNanScalarOrNull, 158
isNegativeNumberOrNaOrNanVector, 159
isNegativeNumberOrNaOrNanVectorOrNull, 161
isNegativeNumberOrNaScalar, 162
isNegativeNumberOrNaScalarOrNull, 163
isNegativeNumberOrNaVector, 164
isNegativeNumberOrNaVectorOrNull, 166
isNegativeNumberScalar, 167
isNegativeNumberScalarOrNull, 168
isNegativeNumberVector, 170
isNegativeNumberVectorOrNull, 171
isNonEmptyStringScalar, 172
isNonEmptyStringScalarOrNull, 173
isNonEmptyStringVector, 175
isNonEmptyStringVectorOrNull, 176
isNonZeroIntegerOrInfScalar, 177
isNonZeroIntegerOrInfScalarOrNull, 179
isNonZeroIntegerOrInfVector, 180
isNonZeroIntegerOrInfVectorOrNull, 181
isNonZeroIntegerOrNanOrInfScalar, 183
isNonZeroIntegerOrNanOrInfScalarOrNull, 184
isNonZeroIntegerOrNanOrInfVector, 185
isNonZeroIntegerOrNanOrInfVectorOrNull, 187
isNonZeroIntegerOrNanScalar, 188
isNonZeroIntegerOrNanScalarOrNull, 189
isNonZeroIntegerOrNanVector, 190
isNonZeroIntegerOrNanVectorOrNull, 192
isNonZeroIntegerOrNaOrInfScalar, 193
isNonZeroIntegerOrNaOrInfScalarOrNull, 194
isNonZeroIntegerOrNaOrInfVector, 196
isNonZeroIntegerOrNaOrInfVectorOrNull, 197

isNonZeroIntegerOrNaOrNanOrInfScalar, 198
isNonZeroIntegerOrNaOrNanOrInfScalarOrNull, 199
isNonZeroIntegerOrNaOrNanOrInfVector, 201
isNonZeroIntegerOrNaOrNanOrInfVectorOrNull, 202
isNonZeroIntegerOrNaOrNanScalar, 203
isNonZeroIntegerOrNaOrNanScalarOrNull, 205
isNonZeroIntegerOrNaOrNanVector, 206
isNonZeroIntegerOrNaOrNanVectorOrNull, 207
isNonZeroIntegerOrNaScalar, 209
isNonZeroIntegerOrNaScalarOrNull, 210
isNonZeroIntegerOrNaVector, 211
isNonZeroIntegerOrNaVectorOrNull, 213
isNonZeroIntegerScalar, 214
isNonZeroIntegerScalarOrNull, 215
isNonZeroIntegerVector, 216
isNonZeroIntegerVectorOrNull, 218
isNonZeroNumberOrInfScalar, 219
isNonZeroNumberOrInfScalarOrNull, 220
isNonZeroNumberOrInfVector, 222
isNonZeroNumberOrInfVectorOrNull, 223
isNonZeroNumberOrNanOrInfScalar, 224
isNonZeroNumberOrNanOrInfScalarOrNull, 225
isNonZeroNumberOrNanOrInfVector, 227
isNonZeroNumberOrNanOrInfVectorOrNull, 228
isNonZeroNumberOrNanScalar, 229
isNonZeroNumberOrNanScalarOrNull, 231
isNonZeroNumberOrNanVector, 232
isNonZeroNumberOrNanVectorOrNull, 233
isNonZeroNumberOrNaOrInfScalar, 235
isNonZeroNumberOrNaOrInfScalarOrNull, 236
isNonZeroNumberOrNaOrInfVector, 237
isNonZeroNumberOrNaOrInfVectorOrNull, 239
isNonZeroNumberOrNaOrNanOrInfScalar, 240
isNonZeroNumberOrNaOrNanOrInfScalarOrNull, 241
isNonZeroNumberOrNaOrNanOrInfVector, 242
isNonZeroNumberOrNaOrNanOrInfVectorOrNull, 244
isNonZeroNumberOrNaOrNanScalar, 245
isNonZeroNumberOrNaOrNanScalarOrNull, 246
isNonZeroNumberOrNaOrNanVector, 248
isNonZeroNumberOrNaOrNanVectorOrNull, 249
isNonZeroNumberOrNaScalar, 250
isNonZeroNumberOrNaScalarOrNull, 251
isNonZeroNumberOrNaVector, 253
isNonZeroNumberOrNaVectorOrNull, 254
isNonZeroNumberScalar, 255
isNonZeroNumberScalarOrNull, 257
isNonZeroNumberVector, 258
isNonZeroNumberVectorOrNull, 259
isNumberOrInfScalar, 261
isNumberOrInfScalarOrNull, 262
isNumberOrInfVector, 263
isNumberOrInfVectorOrNull, 264
isNumberOrNanOrInfScalar, 266
isNumberOrNanOrInfScalarOrNull, 267
isNumberOrNanOrInfVector, 268
isNumberOrNanOrInfVectorOrNull, 270
isNumberOrNanScalar, 271
isNumberOrNanScalarOrNull, 272
isNumberOrNanVector, 273
isNumberOrNanVectorOrNull, 275
isNumberOrNaOrInfScalar, 276
isNumberOrNaOrInfScalarOrNull, 277
isNumberOrNaOrInfVector, 279
isNumberOrNaOrInfVectorOrNull, 280
isNumberOrNaOrNanOrInfScalar, 281
isNumberOrNaOrNanOrInfScalarOrNull, 282
isNumberOrNaOrNanOrInfVector, 284
isNumberOrNaOrNanOrInfVectorOrNull, 285
isNumberOrNaOrNanScalar, 286
isNumberOrNaOrNanScalarOrNull, 288
isNumberOrNaOrNanVector, 289
isNumberOrNaOrNanVectorOrNull, 290
isNumberOrNaScalar, 292
isNumberOrNaScalarOrNull, 293
isNumberOrNaVector, 294
isNumberOrNaVectorOrNull, 295
isNumberScalar, 297
isNumberScalarOrNull, 298

isNumberVector, 299
isNumberVectorOrNull, 300
isPositiveIntegerOrInfScalar, 302
isPositiveIntegerOrInfScalarOrNull, 303
isPositiveIntegerOrInfVector, 304
isPositiveIntegerOrInfVectorOrNull, 306
isPositiveIntegerOrNanOrInfScalar, 307
isPositiveIntegerOrNanOrInfScalarOrNull, 308
isPositiveIntegerOrNanOrInfVector, 309
isPositiveIntegerOrNanOrInfVectorOrNull, 311
isPositiveIntegerOrNanScalar, 312
isPositiveIntegerOrNanScalarOrNull, 313
isPositiveIntegerOrNanVector, 315
isPositiveIntegerOrNanVectorOrNull, 316
isPositiveIntegerOrNaOrInfScalar, 317
isPositiveIntegerOrNaOrInfScalarOrNull, 318
isPositiveIntegerOrNaOrInfVector, 320
isPositiveIntegerOrNaOrInfVectorOrNull, 321
isPositiveIntegerOrNaOrNanOrInfScalar, 322
isPositiveIntegerOrNaOrNanOrInfScalarOrNull, 324
isPositiveIntegerOrNaOrNanOrInfVector, 325
isPositiveIntegerOrNaOrNanOrInfVectorOrNull, 326
isPositiveIntegerOrNaOrNanScalar, 328
isPositiveIntegerOrNaOrNanScalarOrNull, 329
isPositiveIntegerOrNaOrNanVector, 330
isPositiveIntegerOrNaOrNanVectorOrNull, 332
isPositiveIntegerOrNaScalar, 333
isPositiveIntegerOrNaScalarOrNull, 334
isPositiveIntegerOrNaVector, 335
isPositiveIntegerOrNaVectorOrNull, 337
isPositiveIntegerScalar, 338
isPositiveIntegerScalarOrNull, 339
isPositiveIntegerVector, 341
isPositiveIntegerVectorOrNull, 342
isPositiveNumberOrInfScalar, 343
isPositiveNumberOrInfScalarOrNull, 344
isPositiveNumberOrInfVector, 346
isPositiveNumberOrInfVectorOrNull, 347
isPositiveNumberOrNanOrInfScalar, 348
isPositiveNumberOrNanOrInfScalarOrNull, 350
isPositiveNumberOrNanOrInfVector, 351
isPositiveNumberOrNanOrInfVectorOrNull, 352
isPositiveNumberOrNanScalar, 354
isPositiveNumberOrNanScalarOrNull, 355
isPositiveNumberOrNanVector, 356
isPositiveNumberOrNanVectorOrNull, 358
isPositiveNumberOrNaOrInfScalar, 359
isPositiveNumberOrNaOrInfScalarOrNull, 360
isPositiveNumberOrNaOrInfVector, 361
isPositiveNumberOrNaOrInfVectorOrNull, 363
isPositiveNumberOrNaOrNanOrInfScalar, 364
isPositiveNumberOrNaOrNanOrInfScalarOrNull, 365
isPositiveNumberOrNaOrNanOrInfVector, 367
isPositiveNumberOrNaOrNanOrInfVectorOrNull, 368
isPositiveNumberOrNaOrNanScalar, 369
isPositiveNumberOrNaOrNanScalarOrNull, 370
isPositiveNumberOrNaOrNanVector, 372
isPositiveNumberOrNaOrNanVectorOrNull, 373
isPositiveNumberOrNaScalar, 374
isPositiveNumberOrNaScalarOrNull, 376
isPositiveNumberOrNaVector, 377
isPositiveNumberOrNaVectorOrNull, 378
isPositiveNumberScalar, 380
isPositiveNumberScalarOrNull, 381
isPositiveNumberVector, 382
isPositiveNumberVectorOrNull, 384
isStrictlyNegativeIntegerOrInfScalar, 385
isStrictlyNegativeIntegerOrInfScalarOrNull, 386
isStrictlyNegativeIntegerOrInfVector, 387

isStrictlyNegativeIntegerOrInfVectorOrNull, 389
isStrictlyNegativeIntegerOrNanOrInfScalar, 390
isStrictlyNegativeIntegerOrNanOrInfScalarOrNull, 391
isStrictlyNegativeIntegerOrNanOrInfVector, 393
isStrictlyNegativeIntegerOrNanOrInfVectorOrNull, 394
isStrictlyNegativeIntegerOrNanScalar, 395
isStrictlyNegativeIntegerOrNanScalarOrNull, 396
isStrictlyNegativeIntegerOrNanVector, 398
isStrictlyNegativeIntegerOrNanVectorOrNull, 399
isStrictlyNegativeIntegerOrNaOrInfScalar, 400
isStrictlyNegativeIntegerOrNaOrInfScalarOrNull, 402
isStrictlyNegativeIntegerOrNaOrInfVector, 403
isStrictlyNegativeIntegerOrNaOrInfVectorOrNull, 404
isStrictlyNegativeIntegerOrNaOrNanOrInfScalar, 406
isStrictlyNegativeIntegerOrNaOrNanOrInfScalarOrNull, 407
isStrictlyNegativeIntegerOrNaOrNanOrInfVector, 408
isStrictlyNegativeIntegerOrNaOrNanOrInfVectorOrNull, 410
isStrictlyNegativeIntegerOrNaOrNanScalar, 411
isStrictlyNegativeIntegerOrNaOrNanScalarOrNull, 412
isStrictlyNegativeIntegerOrNaOrNanVector, 413
isStrictlyNegativeIntegerOrNaOrNanVectorOrNull, 415
isStrictlyNegativeIntegerOrNaScalar, 416
isStrictlyNegativeIntegerOrNaScalarOrNull, 417
isStrictlyNegativeIntegerOrNaVector, 419
isStrictlyNegativeIntegerOrNaVectorOrNull, 420
isStrictlyNegativeIntegerScalar, 421
isStrictlyNegativeIntegerScalarOrNull, 422
isStrictlyNegativeIntegerVector, 424
isStrictlyNegativeIntegerVectorOrNull, 425
isStrictlyNegativeNumberOrInfScalar, 426
isStrictlyNegativeNumberOrInfScalarOrNull, 428
isStrictlyNegativeNumberOrInfVector, 429
isStrictlyNegativeNumberOrInfVectorOrNull, 430
isStrictlyNegativeNumberOrNanOrInfScalar, 432
isStrictlyNegativeNumberOrNanOrInfScalarOrNull, 433
isStrictlyNegativeNumberOrNanOrInfVector, 434
isStrictlyNegativeNumberOrNanOrInfVectorOrNull, 436
isStrictlyNegativeNumberOrNanScalar, 437
isStrictlyNegativeNumberOrNanScalarOrNull, 438
isStrictlyNegativeNumberOrNanVector, 439
isStrictlyNegativeNumberOrNanVectorOrNull, 441
isStrictlyNegativeNumberOrNaOrInfScalar, 442
isStrictlyNegativeNumberOrNaOrInfScalarOrNull, 443
isStrictlyNegativeNumberOrNaOrInfVector, 445
isStrictlyNegativeNumberOrNaOrInfVectorOrNull, 446
isStrictlyNegativeNumberOrNaOrNanOrInfScalar, 447
isStrictlyNegativeNumberOrNaOrNanOrInfScalarOrNull, 448
isStrictlyNegativeNumberOrNaOrNanOrInfVector, 450
isStrictlyNegativeNumberOrNaOrNanOrInfVectorOrNull, 451

isStrictlyNegativeNumberOrNaOrNanScalar, 452
isStrictlyNegativeNumberOrNaOrNanScalarOrNull, 454
isStrictlyNegativeNumberOrNaOrNanVector, 455
isStrictlyNegativeNumberOrNaOrNanVectorOrNull, 456
isStrictlyNegativeNumberOrNaScalar, 458
isStrictlyNegativeNumberOrNaScalarOrNull, 459
isStrictlyNegativeNumberOrNaVector, 460
isStrictlyNegativeNumberOrNaVectorOrNull, 462
isStrictlyNegativeNumberScalar, 463
isStrictlyNegativeNumberScalarOrNull, 464
isStrictlyNegativeNumberVector, 465
isStrictlyNegativeNumberVectorOrNull, 467
isStrictlyPositiveIntegerOrInfScalar, 468
isStrictlyPositiveIntegerOrInfScalarOrNull, 469
isStrictlyPositiveIntegerOrInfVector, 471
isStrictlyPositiveIntegerOrInfVectorOrNull, 472
isStrictlyPositiveIntegerOrNanOrInfScalar, 473
isStrictlyPositiveIntegerOrNanOrInfScalarOrNull, 474
isStrictlyPositiveIntegerOrNanOrInfVector, 476
isStrictlyPositiveIntegerOrNanOrInfVectorOrNull, 477
isStrictlyPositiveIntegerOrNanScalar, 478
isStrictlyPositiveIntegerOrNanScalarOrNull, 480
isStrictlyPositiveIntegerOrNanVector, 481
isStrictlyPositiveIntegerOrNanVectorOrNull, 482
isStrictlyPositiveIntegerOrNaOrInfScalar, 484
isStrictlyPositiveIntegerOrNaOrInfScalarOrNull, 485
isStrictlyPositiveIntegerOrNaOrInfVector, 486
isStrictlyPositiveIntegerOrNaOrInfVectorOrNull, 488
isStrictlyPositiveIntegerOrNaOrNanOrInfScalar, 489
isStrictlyPositiveIntegerOrNaOrNanOrInfScalarOrNull, 490
isStrictlyPositiveIntegerOrNaOrNanOrInfVector, 491
isStrictlyPositiveIntegerOrNaOrNanOrInfVectorOrNull, 493
isStrictlyPositiveIntegerOrNaOrNanScalar, 494
isStrictlyPositiveIntegerOrNaOrNanScalarOrNull, 495
isStrictlyPositiveIntegerOrNaOrNanVector, 497
isStrictlyPositiveIntegerOrNaOrNanVectorOrNull, 498
isStrictlyPositiveIntegerOrNaScalar, 499
isStrictlyPositiveIntegerOrNaScalarOrNull, 500
isStrictlyPositiveIntegerOrNaVector, 502
isStrictlyPositiveIntegerOrNaVectorOrNull, 503
isStrictlyPositiveIntegerScalar, 504
isStrictlyPositiveIntegerScalarOrNull, 506
isStrictlyPositiveIntegerVector, 507
isStrictlyPositiveIntegerVectorOrNull, 508
isStrictlyPositiveNumberOrInfScalar, 510
isStrictlyPositiveNumberOrInfScalarOrNull, 511
isStrictlyPositiveNumberOrInfVector, 512
isStrictlyPositiveNumberOrInfVectorOrNull, 514
isStrictlyPositiveNumberOrNanOrInfScalar, 515
isStrictlyPositiveNumberOrNanOrInfScalarOrNull, 516

isStrictlyPositiveNumberOrNanOrInfVector, 517
isStrictlyPositiveNumberOrNanOrInfVectorOrNull, 519
isStrictlyPositiveNumberOrNanScalar, 520
isStrictlyPositiveNumberOrNanScalarOrNull, 521
isStrictlyPositiveNumberOrNanVector, 523
isStrictlyPositiveNumberOrNanVectorOrNull, 524
isStrictlyPositiveNumberOrNaOrInfScalar, 525
isStrictlyPositiveNumberOrNaOrInfScalarOrNull, 526
isStrictlyPositiveNumberOrNaOrInfVector, 528
isStrictlyPositiveNumberOrNaOrInfVectorOrNull, 529
isStrictlyPositiveNumberOrNaOrNanOrInfScalar, 530
isStrictlyPositiveNumberOrNaOrNanOrInfScalarOrNull, 532
isStrictlyPositiveNumberOrNaOrNanOrInfVector, 533
isStrictlyPositiveNumberOrNaOrNanOrInfVectorOrNull, 534
isStrictlyPositiveNumberOrNaOrNanScalar, 536
isStrictlyPositiveNumberOrNaOrNanScalarOrNull, 537
isStrictlyPositiveNumberOrNaOrNanVector, 538
isStrictlyPositiveNumberOrNaOrNanVectorOrNull, 540
isStrictlyPositiveNumberOrNaScalar, 541
isStrictlyPositiveNumberOrNaScalarOrNull, 542
isStrictlyPositiveNumberOrNaVector, 543
isStrictlyPositiveNumberOrNaVectorOrNull, 545
isStrictlyPositiveNumberScalar, 546
isStrictlyPositiveNumberScalarOrNull, 547
isStrictlyPositiveNumberVector, 549
isStrictlyPositiveNumberVectorOrNull, 550
isStringScalar, 551
isStringScalarOrNull, 552
isStringVector, 554
isStringVectorOrNull, 555
isZeroOrNanScalar, 556
isZeroOrNanScalarOrNull, 557
isZeroOrNanVector, 559
isZeroOrNanVectorOrNull, 560
isZeroOrNaOrNanScalar, 561
isZeroOrNaOrNanScalarOrNull, 562
isZeroOrNaOrNanVector, 564
isZeroOrNaOrNanVectorOrNull, 565
isZeroOrNaScalar, 566
isZeroOrNaScalarOrNull, 567
isZeroOrNaVector, 569
isZeroOrNaVectorOrNull, 570
isZeroScalar, 571
isZeroScalarOrNull, 572
isZeroVector, 573
isZeroVectorOrNull, 575