Package 'easy.utils'

July 22, 2025

Type Package

Title Frequently Used Functions for Easy R Programming

Version 0.1.0

Description Some utility functions for validation and data manipulation. These functions can be helpful to reduce internal codes everywhere in package development.

Depends R (>= 4.1.0), methods

Imports dplyr, fastmatch, rlang, scales

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/ycli1995/easy.utils

BugReports https://github.com/ycli1995/easy.utils/issues

RoxygenNote 7.3.1

NeedsCompilation no

Author Yuchen Li [aut, cre]

Maintainer Yuchen Li <ycli1995@outlook.com>

Repository CRAN

Date/Publication 2025-02-19 10:00:02 UTC

Contents

`checkAlignedDims` *Check whether some dimensions of two arrays are aligned*

Description

Check whether some dimensions of two arrays are aligned

Usage

```
checkAlignedDims(
  incoming,
  reference,
  align.dims,
  in.name = NULL,
  ref.name = NULL,
  withDimnames = FALSE
)
```

Arguments

`incoming`	The array-like object to check
`reference`	The array-like object to be aligned with
`align.dims`	A integer vector indicating which dimensions of `reference` should be used for alignment. The length must be equal to the dimension numbers of `incoming`
`in.name`	The name of `incoming`. Only use for verbose.
`ref.name`	The name of `reference`. Only use for verbose.
`withDimnames`	Logical. Whether to also align the dimension names.

Details

Some examples for `align.dims`:

- `c(1, 1)`: The dim[1] of `incoming` must align with the dim[1] of `reference`, and the dim[2] of `incoming` must align with the dim[1] of `reference`.
- `c(2, 1)`: The dim[1] of `incoming` must align with the dim[2] of `reference`, and the dim[2] of `incoming` must align with the dim[1] of `reference`.
- `c(NA, 1)`: The dim[1] of `incoming` doesn't need to align with any dimension of `reference`, but the dim[2] of `incoming` must align with the dim[1] of `reference`.
- `c(2, NA)`: The dim[1] of `incoming` must align with the dim[2] of `reference`, but the dim[2] of `incoming` doesn't need to align with any dimension of `reference`.

Value

If any dimension is not aligned, raise an error.

Examples

```
# Get some expression matrices ----
exp1 <- matrix(0, 10, 20)
colnames(exp1) <- paste0("cell_", 1:ncol(exp1))
rownames(exp1) <- paste0("gene_", 1:nrow(exp1))

exp2 <- matrix(0, 10, 15)
colnames(exp2) <- paste0("cell_", 1:ncol(exp2))
rownames(exp2) <- paste0("gene_", 1:nrow(exp2))

exp3 <- matrix(0, 10, 20)
colnames(exp3) <- paste0("c_", 1:ncol(exp3))
rownames(exp3) <- paste0("g_", 1:nrow(exp3))

# Get some PCA embbeding matrices ----
pca1 <- matrix(0, 10, 5)
rownames(pca1) <- paste0("cell_", 1:nrow(pca1))
colnames(pca1) <- paste0("PC_", 1:ncol(pca1))

pca2 <- matrix(0, 20, 5)
rownames(pca2) <- paste0("cell_", 1:nrow(pca2))
colnames(pca2) <- paste0("PC_", 1:ncol(pca2))

pca3 <- matrix(0, 20, 5)
rownames(pca3) <- paste0("c_", 1:nrow(pca3))
colnames(pca3) <- paste0("PC_", 1:ncol(pca3))

# Error: The Dim 2 of exp1 is not aligned with the Dim 2 of exp2!
try(checkAlignedDims(exp2, exp1, c(1, 2)))

checkAlignedDims(exp3, exp1, c(1, 2))

# Error: The Dim 1 of exp3 is not aligned with the Dim 1 of exp1!
try(checkAlignedDims(exp3, exp1, c(1, 2), withDimnames = TRUE))

checkAlignedDims(exp3, exp1, c(NA, 2)) # Don't check the rows of exp3

# Error: The Dim 2 of exp3 is not aligned with the Dim 2 of exp1!
try(checkAlignedDims(exp3, exp1, c(NA, 2), withDimnames = TRUE))

# Error: The Dim 1 of pca1 is not aligned with the Dim 2 of exp1!
# Don't check the columns of pca1
try(checkAlignedDims(pca1, exp1, c(2, NA)))

checkAlignedDims(pca2, exp1, c(2, NA))
checkAlignedDims(pca2, exp1, c(2, NA), withDimnames = TRUE)
checkAlignedDims(pca3, exp1, c(2, NA))

# Error: The Dim 1 of pca3 is not aligned with the Dim 2 of exp1!
try(checkAlignedDims(pca3, exp1, c(2, NA), withDimnames = TRUE))
```

checkSameLength *Check whether the lengths of input objects are equal*

Description

Check whether the lengths of input objects are equal

Usage

```
checkSameLength(...)
```

Arguments

...	R objects to be compared

Value

TRUE or FALSE

chunkPoints *Generate chunk points*

Description

Unexported helper function ChunkPoints from **Seurat**. This can be quite useful when user needs to chunk some operations.

Usage

```
chunkPoints(dsize, csize)
```

Arguments

dsize	How big is the data being chunked
csize	How big should each chunk be

Value

A 2 x N matrix where each column is a chunk. The first row contains start points, and the second row contains end points.

References

https://github.com/satijalab/seurat/blob/763259d05991d40721dee99c9919ec6d4491d15e/R/utilities.R#L1699

Examples

```
### Split an index vector with 15273 elements into chunks, each of which has
### 3000 elements.
chunkPoints(15273, 3000)
```

`fastIntersect` *A fast version of base::intersect()*

Description

A fast version of base::intersect()

Usage

```
fastIntersect(x, y, keep.duplicated = FALSE)
```

Arguments

x, y	Vectors to be compared.
keep.duplicated	Whether or not to keep duplicated elements in x

Value

A vector of a common mode.

References

https://stackoverflow.com/questions/72631297/speed-up-setdiff-intersect-union-operations-on-vectors

See Also

intersect

Examples

```
x <- sample(LETTERS, 12)
y <- sample(LETTERS, 12)
fastIntersect(x, y)
```

fetchColnames *Fetch column names exists in the data object*

Description

Fetch column names exists in the data object

Usage

```
fetchColnames(object, query)
```

Arguments

object	Any object that has implemented colnames(object).
query	Column names to check.

Value

An update query where only entries existing in colnames(object) are kept. If no any query was found, raise an error.

identicalNoAttr *Equality testing with some attributes ignored*

Description

A wrapper for function identical. Some attributes of the two objects can be ignored when testing.

Usage

```
identicalNoAttr(x, y, ignore.attrs = NULL, ...)

identicalFMatch(x, y, ...)
```

Arguments

x, y	Any R objects.
ignore.attrs	Names of attributes in ‘x‘ and ‘y‘. The selected attributes will be removed before testing. Default is ‘NULL‘ (keep all attributes)
...	Arguments passed to identical.

Details

‘identicalFMatch‘ is a wrapper for ‘identicalNoAttr‘, where ‘ignore.attrs‘ is set to ‘".match.hash"‘. This function is helpful to test two vectors after using fmatch, which add external hash tables to the compared vectors.

Value

A single logical value ('TRUE' or 'FALSE'), same as identical.

Examples

```
x1 <- LETTERS[1:10]
x2 <- sample(x1, 5)
x3 <- x1[fastmatch::fmatch(x2, x1)]
identical(x3, x2)  ## TRUE, but x1 has the '.match.hash' attribute now.

identical(LETTERS[1:10], x1)  ## FALSE
identicalFMatch(x3, x2)  ## TRUE
```

---

isValidCharacters *Check valid characters*

Description

Check if input characters are valid (neither NA nor "")

Usage

```
isValidCharacters(x)
```

Arguments

x	A vector, matrix or list

Value

A logical vector

Examples

```
isValidCharacters(c("a", "", "b"))
isValidCharacters(c("a", NA, "b"))
```

pasteFactors *Paste two factor vectors*

Description

Paste two factors and re-assign the levels

Usage

```
pasteFactors(x, y, collapse = "_")
```

Arguments

x, y	Factor vectors
collapse	A character string to separate the x and y.

Value

A new factor vector

Examples

```
x <- factor(c(rep("A", 10), rep("B", 10)), levels = c("A", "B"))
y <- factor(c(rep("a", 5), rep("b", 15)), levels = c("a", "b"))
pasteFactors(x, y)
```

replaceEntries *Replace entries according to a mapping list*

Description

Replace entries according to a mapping list

Usage

```
replaceEntries(x, map, ...)

## S4 method for signature 'vector,list'
replaceEntries(x, map, ...)
```

Arguments

x	An R vector
map	A named list representing one-to-one or one-to-many mappings. Normally, each name represents a new value, and each element contain the old value(s) to be replaced.
...	Arguments passed to other methods.

Value

A updated x

Examples

```
set.seed(1234)
fact <- factor(c("A", "A", "B", "A", "B", "C", "D", "E", "D"))
map <- list("a" = c("B", "e")) ## Turn all "B" and "E" into "a"
replaceEntries(fact, map)
```

---

unlistMap *Unlist a mapping list into a named vector*

Description

Function to unlist a one-to-one or one-to-many 'key-value' `list` into a named `vector`. Useful for batched replacement of vector elements.

Usage

```
unlistMap(map, keep.unique = TRUE)
```

Arguments

`map`	A named list. Each element must be a vector.
`keep.unique`	Whether or not to remove elements with duplicated names from the output vector.

Value

A named vector whose names are original values in `map`, and elements are keys of `map`

Examples

```
map <- list(X = c("a", "b"), Y = c("c", "d"))
unlistMap(map)

map <- list(X = c("a", "b", "c"), Y = c("c", "d"))
unlistMap(map)
unlistMap(map, keep.unique = FALSE)
```

validation-matrix_dimensions
Validation functions for the dimensions of matrix-like objects

Description

Functions to check whether a matrix-like object has expected dimension numbers or names.

Usage

```
validMatDims(mat, nrow = NULL, ncol = NULL)

validMatDimnames(
  mat,
  row.names = NULL,
  col.names = NULL,
  dup.rownames = FALSE,
  dup.colnames = FALSE
)
```

Arguments

`mat`	A matrix-like object
`nrow`	Expect how many rows in ‘mat‘.
`ncol`	Expect how many columns in ‘mat‘.
`row.names`	Expected row names for ‘mat‘.
`col.names`	Expected column names for ‘mat‘.
`dup.rownames, dup.colnames`	Whether or not to allow duplicated dimension names in ‘mat‘.

Value

If all the validations are passed, return invisible ‘NULL‘.

Examples

```
mat1 <- matrix(0, 3, 5)
validMatDims(mat1, 3, 5)

## Check dimnames
mat1 <- matrix(0, 3, 5)
rownames(mat1) <- letters[1:3]
colnames(mat1) <- LETTERS[1:5]
try(validMatDimnames(mat1, row.names = letters[2:4]))  ## Error
rownames(mat1) <- c("A", "B", "A")
try(validMatDimnames(mat1, row.names = letters[2:4]))  ## Error
```

verboseMsg *Simple verbose message wrapper*

Description

Simple verbose message wrapper

Usage

```
verboseMsg(..., verbose = NULL)
```

Arguments

...	Pass to message
verbose	Whether or not to show the message. If is NULL, will search verbose variable in parent.frame.

Value

Print the progress to console when verbose is TRUE.

Index