Package: easy.utils (via r-universe)

September 15, 2024

Type Package

Title Frequently Used Functions for Easy R Programming

Version 0.0.4

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

Depends R (>= 4.1.0), methods

Imports dplyr, fastmatch, Polychrome, rlang, scales

Suggests randomcoloR

License MIT + file LICENSE

Encoding UTF-8

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

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

LazyData true

RoxygenNote 7.2.3

Repository https://ycli1995.r-universe.dev

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

RemoteRef HEAD

RemoteSha cc8ea784a4fde483be840650a9cbb1003b65a7ec

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))
```

---

checkColorMap *Check color map for a factor*

Description

Function to make sure that all levels of a factor map to distinct colors.

Usage

```
checkColorMap(x, colors = NULL, ...)

## Default S3 method:
checkColorMap(x, colors = NULL, ...)
```

Arguments

x	An R object contains the factor vector to be checked.
colors	A named vector, whose names are factor levels and values are colors. If is NULL, just generate colors with setColor. Otherwise, it first generates colors for each level, then replaces those with names mapping to colors.
...	Arguments passed to other methods.

Value

An updated colors vector, whose names are identical to the levels.

Examples

```
## Assign colors for a character or factor vector.
xx <- sample(LETTERS, 10)
cols <- setColor(xx)

## Ensure each level in 'xx' get a color
checkColorMap(xx, cols)
```

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.

getDiscreteColors *Generate palettes of distinct colors*

Description

Generate palettes of distinct colors

Usage

```
getDiscreteColors(
  n,
  pal = NULL,
  is.extend = TRUE,
  random = c("no", "randomColor", "distinctColorPalette", "Polychrome"),
  seed = 1234,
  verbose = FALSE,
  ...
)

setColor(x, pal = NULL, ...)
```

Arguments

n	How many colors do we need?
pal	Name of the palette to use. Use `names(pal_discrete)` to get all palette names
is.extend	When `n > length(pal_discrete[[pal]])`, whether or not to extend the colors with `colorRampPalette`.
random	Choose a method to generate random colors. Default is "no".
seed	Seed for random colors.
verbose	Show progress messages.
...	Extra parameters passed to other functions depending on `random`: • `randomColor()` or `distinctColorPalette()`. This requires manual installation of **randomcoloR** • `createPalette` from **Polychrome**
x	A factor to use colors.

Value

A vector with `n` colors. For `setColor`, also set names as factor levels.

Examples

```
getDiscreteColors(10)
getDiscreteColors(
  10,
  random = "Polychrome",
  seedcolors = scales::hue_pal()(4)
)

## Assign colors for a character or factor vector.
xx <- sample(LETTERS, 10)
cols <- setColor(xx)
```

`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"))
```

---

pal_discrete *A palette list for distinct colors.*

Description

A palette list for distinct colors.

Usage

```
pal_discrete
```

Format

An object of class `list` of length 20.

---

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)
```

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