Package: inSilecoMisc (via r-universe)

August 26, 2024

Title inSileco Miscellaneous Functions

Date 2022-09-09

Version 0.7.0.9000

Description A set of miscellaneous R functions written by our inSileco group.

Depends R (>= 3.0)

License GPL (>= 2)

Imports cli, crayon, glue, methods, knitr (>= 1.22), rmarkdown, utils, yaml

Suggests datasets, graphics, testthat

SystemRequirements pandoc (>= 1.12.3) - http://pandoc.org

URL https://github.com/inSileco/inSilecoMisc, http://insileco.github.io/inSilecoMisc

BugReports https://github.com/inSileco/inSilecoMisc/issues

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.2.1

Roxygen list(markdown = TRUE)

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

RemoteUrl https://github.com/inSileco/inSilecoMisc

RemoteRef HEAD

RemoteSha 1b4d08555e916dd5a73f4b249d4c8573c5129509

Contents

`adjustStrings` *Adjust the size of character strings*

Description

Adjust the size of character strings by adding extra characters. It allows to constraint the size of the string or to add a specific number of extra characters.

Usage

```
adjustStrings(
  x,
  n,
  extra = 0,
  align = c("right", "left", "center"),
  add = FALSE
)
```

Arguments

x	a character vector, or a vector to be coerced to a character vector.
n	a positive integer indicating the size of character strings to be created or added.
extra	a character vector, or a vector to be coerced to a character vector that will be (partially) added to produced a string of a specific length.
align	either "right", "left" or "center". If "right", then strings will be right-aligned and so extra's characters will be added to the left. "right" is the opposite of "left" and if align = "center", then extra characters will be split in half and added on both sides.
add	a logical should extra be added, in which case n characters will be added to input strings.

Details

This function was originally created to help getting a fixed number of digits when naming files. The current version is more general, it allows to add any string before or after to adjust the size. Not that if a character is longer than expected, it will be adequately cut off.

Value

A character vector of the concatenated characters.

See Also

base::sprintf()

Examples

```
paste0('myfilename', adjustStrings(c(1:2,10,100), 3))
adjustStrings('# Comment ', 20, '#', align = "left")
```

---

aggregateCol *Aggregate columns*

Description

aggregateCol() performs summary statistics on a specific subset of columns.

Usage

```
aggregateCol(data, grp, names_aggreg = NULL, FUN = sum, ...)
```

Arguments

data	a data frame.
grp	an integer vector, one value per data column. Integers are used to group data columns, column with the same integer will be pooled together. Any integer is a valid group, 0 indicates that the column must be unchanged, NA removes columns.
names_aggreg	column names for aggregated columns.
FUN	a function to be applied to all groups of columns.
...	further arguments to be passed to FUN.

Details

grp is an integer vector, whose length is the number of columns of the input data frame, is used to assign a group to each column that explicit the sub-setting. Furthermore columns can be either kept as is using 0 or discarded using NA.

Value

A data frame with the grouped columns.

See Also

stats::aggregate() base::split

Examples

```
mat1 <- matrix(1:70,10)
grp1 <- c(NA,0, 1, 1, 2, 2, 2)
aggregateCol(mat1, grp1, FUN = mean)
```

---

applyString *Apply a function on character strings*

Description

Apply a function on a given set of elements of a character string.

Usage

```
applyString(x, FUN, pos = NULL, pattern = NULL)
```

Arguments

x	a character vector, or a vector to be coerced to a character vector.
FUN	the function to be applied, see base::lapply().
pos	a vector indicating the elements position.
pattern	a pattern see base::gregexpr().

Value

A character vector.

Note

In case both pos or pattern, the latter is ignored.

Examples

```
applyString('cool',  pos = 1:2, FUN = toupper)
applyString(c('cool', 'pro'),  pattern = 'o', FUN = toupper)
```

---

assignClass2df *Assign a class to columns*

Description

Assign a class to a set of column of a data frame. This function is designed to ease the changes of column's class of a given data frame.

Usage

```
assignClass2df(x, colid, cls)
```

Arguments

x	a data frame or a R object to be coerced into a data frame.
colid	the identity of columns for which class are ti be changed.
cls	a character vector containing the classes' names to be used in the same order as colid. By default cls is repeated until its size equals colid's size.

Value

A data frame for which columns have the required classes.

Examples

```
df1 <- matrix(signif(runif(20),4), ncol=2)
df2 <- assignClass2df(df1, 2, 'character')
str(df1)
str(df2)
```

assignIds *Assign an id to a list of characters or factors*

Description

This function aims to assign an id to factor or similar character strings. Regarding factors, `as.integer()` does such task but the order obtained may differ.

Usage

```
assignIds(x, alphabetical = FALSE)
```

Arguments

`x`	an R object to be coerced into character type.
`alphabetical`	a logical indicating whether an alphabetical sorting must be applied.

Value

A vector of Ids.

Examples

```
assignIds(list(2,'f', 'd', 'f'))
```

categorize *Assign categories*

Description

Assigns a category to each element of a vector for a given set of threshold values.

Usage

```
categorize(x, categ, lower = FALSE)
```

Arguments

`x`	a numeric, complex, character or logical vector.
`categ`	a set of threshold values used to assign categories.
`lower`	a logical. If `TRUE` threshold values (i.e. values within `categ`) belongs to the lower category rather than the upper (default behavior).

Value

A vector of categories assigned.

Examples

```
categorize(stats::runif(40), categ=c(0.5,0.75))
categorize(LETTERS[1:5], categ='C')
categorize(LETTERS[1:5], categ='C',  lower=TRUE)
categorize(LETTERS[floor(5*stats::runif(20))+1], categ=LETTERS[1:5], lower=TRUE)
```

---

commaAnd *Paste element and add elememt separators.*

Description

Paste element and add elememt separators.

Usage

```
commaAnd(x, comma = ", ", and = " and ")
```

Arguments

x	vector of to be coerced to character strings (see paste()).
comma	element separator.
and	last element separator (for 3 elements in x and more).

Examples

```
commaAnd(c("Judith", "Peter", "Rebecca"))
```

---

dfTemplate *Create a data frame from scratch or based on one or two data frames*

Description

This function handles the creation of data frames based on intuitive parameters. It was originally designed to make row binding easier when columns differs among data frame by creating data frames with the same columns.

Usage

```
dfTemplate(cols, nrows = 1, col_classes = NULL, fill = NA)

dfTemplateMatch(x, y, yonly = FALSE, order = FALSE, ...)
```

Arguments

cols	either a number of column or a vector of character used as columns names of the data frame to be returned.
nrows	row number.
col_classes	vector of column classes for the desired data frame. By default, the class is determined by fill.
fill	character or number used to fill out the columns. Default is NA.
x	a data frame.
y	a data frame or a vector of strings use to specifies column names to be included in the data frame.
yonly	a logical. Should only y (or the names(y)) be used for the data frame to be returned? Default is set to FALSE meaning that both the names of x and the names of y are used.
order	a logical. Should column of the output data frame be ordered according to the template. Not that if there are more columns in x that are not in y, then if order = TRUE column of y will be added first (on the left of the output data frame).
...	further arguments to be passed to dfTemplate().

Value

Returns a data frame with the desired characteristics.

Functions

- dfTemplateMatch(): Returns a data frame that includes all columns specifies in y.

References

https://insileco.github.io/2019/02/03/creating-empty-data-frames-with-dftemplate-and-dftemplatemat

Examples

```
dfTemplate(5, 2)
dfTemplate(5, 2, col_classes = "character")
dfA <- data.frame(col1 = c(1, 2), col2 = LETTERS[1:2])
dfB <- data.frame(col4 = c(1, 2), col2 = LETTERS[1:2])
dfTemplateMatch(dfA, c("col4", "col2"))
dfTemplateMatch(dfA, c("col4", "col2"), yonly = TRUE)
dfTemplateMatch(dfA, dfB, yonly = TRUE, order = TRUE)
```

duplicateRow *Duplicates elements of a data frame*

Description

Duplicates rows and columns of a given a data frame.

Usage

```
duplicateRow(x, id.el = 1, times = 1, append = FALSE)

duplicateCol(x, id.el = 1, times = 1, append = FALSE)
```

Arguments

x	a data frame.
id.el	identity of the elements to be duplicated.
times	number of times elements are duplicated. Could be a vector of the same length as id.el.
append	A logical. If TRUE, duplicated elements will be appended to the data frame otherwise duplicated elements remain next to their parent. Non-existing columns cannot be duplicated while non-existing rows can and produce NA.

Value

returns a data frame with duplicated rows.

Functions

- duplicateCol(): returns a data frame with duplicated columns.

Examples

```
data(iris, package = "datasets")
iris2 <- duplicateRow(iris, id.el = 1:50, times = 2)
iris3 <- duplicateCol(
iris,
id.el = c("Petal.Length", "Petal.Width"),
times = c(1, 2),
append = TRUE)
```

findThem *Find values in a given vector*

Description

Given a set of values and a vector where the values must be found, findThem records the matched values and the position in the vector where the values have been searched for.

Usage

```
findThem(what, where, todf = FALSE, reportnomatch = FALSE)
```

Arguments

what	a vector a values to be searched for.
where	a vector where values will be searched on.
todf	a logical. Should the output object must be a data frame?
reportnomatch	a logical. If TRUE, values without match are reported #' in the data frame with a NA. Only available if todf is TRUE (default is FALSE).

Value

A list indicating matched positions for each elements of what. If no match is found then NA is returned. If todf is TRUE then a three-columns data frame is returned including values and positions in both what and where vectors.

See Also

which()

Examples

```
x <- stats::rpois(1000,10)
findThem(c(10,4,100), x)
findThem(c(10,4,100), x, todf=TRUE)
findThem(c(10,4,100), x, todf=TRUE, reportnomatch=TRUE)
```

gaussianShape *Flexible bell-shaped function*

Description

The gaussianShape() function computes a personnalisable Gaussian function. Default values are the same as in stats::dnorm().

Usage

```
gaussianShape(x, optx = 0, opty = 1/sqrt(2 * pi), width = 1, pow = 2)
```

Arguments

x	a numeric vector.
optx	x-values at which the maximum is reached.
opty	extremum value.
width	width of the bell.
pow	a real number.

Value

A vector containing values standing for categories into which elements of x have fallen.

See Also

stats::dnorm()

Examples

```
gaussianShape(0)
plot(gaussianShape(1:1000, 500, 2, 250, pow=5), type='l')
```

getDigits *Extract digits from a character vector*

Description

This function extracts all digits found in character vector and returns them as a list.

Usage

```
getDigits(x, collapse = NULL)
```

Arguments

x	a character vector where digits are sought, or an object which can be coerced by `as.character` to a character vector.
collapse	an optional character string to separate the results (see [base::paste()).

Value

A list of digits.

Examples

```
getDigits('txt012-34')
```

---

getExtension *Extract file name, extension and basename ofrom a path.é*

Description

Extract file name, extension and basename ofrom a path.é

Usage

```
getExtension(x, sep = .Platform$file.sep)

getFilename(x, sep = .Platform$file.sep)

getName(x, sep = .Platform$file.sep)

getBasename(x, sep = .Platform$file.sep)

getLocation(x, sep = .Platform$file.sep)

getDetails(x, sep = .Platform$file.sep)
```

Arguments

x	a character string. Note that `getDetails()` also supports vector of paths.
sep	file separator, default used the platform-specific file separator see `.Platform()`.

Details

For more functionalities, have a look at package fs.

Value

Extract the file extension a character string (e.g. a path).

Functions

- `getFilename()`: Extract the file name from a character string, that is the base name and the file extension
- `getName()`: Extract the name of a file or a directory from a character string.
- `getBasename()`: Extract the base name of a file from a character string.
- `getLocation()`: Extract the location of the parent folfder from a character string.
- `getDetails()`: Return a data frame with basic information for all elements of a vector of paths.

Examples

```
getExtension("path1/path2/foo.R")
getFilename("path1/path2/foo.R")
getBasename("path1/path2/foo.R")
getLocation("path1/path2/foo.R")
getDetails("path1/path2/foo.R")
getDetails(list.files(recursive = TRUE))
getExtension("foo.R")
getBasename("foo.R")
```

---

keepWords *Keep a selection of words or letters*

Description

Select words or letters based on their position in character strings.

Usage

```
keepWords(
  str,
  slc = 1,
  collapse = " ",
  na.rm = FALSE,
  split_words = "[[:punct:][:space:]]+"
)

keepLetters(
  str,
  slc = 1,
  collapse = "",
  na.rm = FALSE,
  rm_punct = "[[:punct:][:space:]]+"
)

keepInitials(str, split_words = "[\n\t\r\f\b[:punct:] ]+", collapse = "")

wordCount(str, split_words = "[[:punct:][:space:]]+")
```

Arguments

str	an input character vector (or a list) from which words will be extracted.
slc	a vector of integer indicating the selected positions of the words (or letters) to be kept.
collapse	character string used to separate selected words (or letters), if NULL, then selection is not collapsed and a list is returned.
na.rm	a logical. Should missing values be removed?
split_words	a character string containing a regular expression used to split words.
rm_punct	a character string containing a regular expression used to remove punctuation characters.

Value

A vector (or a list) of the selected words.

Functions

- keepLetters(): A vector (or a list) of the selected letters.
- keepInitials(): A vector (or a list) of initials.
- wordCount(): A vector of the number of words for every character strings passed as an input.

See Also

strsplit()

Examples

```
keepWords(loremIpsum(), 1:3)
keepWords(c(loremIpsum(),'Another character string!'), slc = c(1,4))
keepWords(c(loremIpsum(),'A second character string.'), slc = c(1,4),
 na.rm = TRUE, collapse = '/')
strex <- c('Lorem ipsum', 'dolor sit', ' amet;')
keepLetters(strex, c(1,4))
keepLetters(strex, c(1,4), collapse = "")
keepInitials("National Basketball Association")
wordCount(c("two words!", "... and three words"))
wordCount(loremIpsum())
```

`logistic` *Logistic functions*

Description

The `logisic` function describe the classical logistic function,

Usage

```
logistic(x, yneg = -1, ypos = 1, lambda = 1, pow = 1)

logistic2(x, yneg = -1, ypos = 1, lambda = 1, pow = 1, yzer = 0)
```

Arguments

`x`	a numerical vector.
`yneg`	asymptotic values when x tends to `-Inf`.
`ypos`	asymptotic values when y tends to `-Inf`.
`lambda`	scalar coefficient.
`pow`	x exponent.
`yzer`	values (for `logistic2` only).

Details

The classic logistic equation is:

$$f(x) = \frac{ypos - yneg}{1 + e^{-\lambda x^{pow}}}$$

A slightly different version is:

$$f(x) = yneg + \frac{1}{\frac{1}{ypos-yneg} + (\frac{1}{yzer-yneg} - \frac{1}{ypos-yneg})e^{-\lambda x^{pow}}}$$

Value

A numeric vector.

Functions

- `logistic2()`: A slightly different logistic function.

Source

https://en.wikipedia.org/wiki/Logistic_functionwikipedia.org/wiki/Logistic_function

loremIpsum *Lorem ipsum.*

Description

Returns a common form of the Lorem ipsum text an optionally a subset of it.

Usage

```
loremIpsum(n = NULL)
```

Arguments

n	the number of words to be kept.

Value

A character string with the desired piece of lorem ipsum.

References

https://en.wikipedia.org/wiki/Lorem_ipsum

Examples

```
loremIpsum(5)
```

meanAlong *Compute the mean along a vector*

Description

This function is a simple moving window function.

Usage

```
meanAlong(vec, n)
```

Arguments

vec	a vector of numeric.
n	an integer indicating the size of the window.

Examples

```
meanAlong(1:10, 2)
```

msgInfo *Pre-formatted message functions*

Description

Convenient wrappers around `message()` to sent styled notices to the user while a function/script is being executed.

Usage

```
msgInfo(..., appendLF = TRUE)

msgError(..., appendLF = TRUE)

msgSuccess(..., appendLF = TRUE)

msgWarning(..., appendLF = TRUE)
```

Arguments

`...`	text to be passed to `paste()`.
`appendLF`	a logical. Should messages given as a character string have a new line appended ?

Value

Reports an info.

Functions

- `msgError()`: Reports an Error.
- `msgSuccess()`: Reports a success.
- `msgWarning()`: Reports a warning.

Note

All of these functions call `message()`, so for any function `FUN()` using #' them, `suppressMessages(FUN())` is sufficient to mute all messages.

See Also

`cli::symbol()` `message()`

Examples

```
msgInfo("computing")
msgSuccess("done")
msgError("you got it wrong")
msgWarning("be careful")
```

`multiMatch` *Multiple match*

Description

Seek all elements matching a given pattern in a character strings.

Usage

```
multiMatch(text, pattern)
```

Arguments

`text`	a character vector where matches are sought, or an object which can be coerced by as.character to a character vector, see `regexec`.
`pattern`	character string containing a regular expression to be matched in the given character vector, see `regexec`.

Details

`multiMatch` is essentially a wrapper around `base::regexec()` and `base::regmatches()` to find more than one match.

Value

A vector of character strings matching `pattern` argument.

Examples

```
multiMatch(loremIpsum(), " [[:alnum:]]{3} ")
```

packagesUsed *Get a data frame of package and their installed version.*

Description

Get a data frame of package and their installed version.

Usage

```
packagesUsed(vc_pkg)
```

Arguments

vc_pkg	a vector of string characters including package names to be added.

Value

A data frame of two columns. First is the name of the package; second the version installed.

Examples

```
packagesUsed(c('utils', 'methods'))
```

readYamlHeader *Reads YAML headers*

Description

Reads YAML headers (typically found in Markdown files).

Usage

```
readYamlHeader(con)
```

Arguments

con	connection object or a character string.

Value

A named list, see `yaml::yaml.load()` for more details.

rename *Rename object*

Description

This function finds old names and replace them with new ones.

Usage

```
rename(x, old, new)
```

Arguments

x	an object from which names can be extracted see names().
old	a vector of character strings of old names.
new	a vector of character strings of new names.

Examples

```
tb <- data.frame(var1 = 2, var2 = "B")
rename(tb, "var1", "uptake")
rename(tb, c("var1", "var2"), c("uptake", "type"))
```

scaleWithin *Scale a set of values*

Description

This function maps a set of values to a range of integers (from 1 to a positive integer passed as an argument).

Usage

```
scaleWithin(x, n = 100, mn = min(x), mx = max(x))
```

Arguments

x	a numeric object of type numeric or coercible as one.
n	an integer designing The number of scaled values (between 1 and n).
mn	minimum value (mn and lower values are set to 1).
mx	maximum value (mx and higher values are set to n).

Details

This function was originally created to ease the creation of custom color scales. In such context, it is common practice to define a minium and a maximum for the set of values to be displayed values as well as a number of color to be used in the color scale (`n`). This is exactly what `scaleWithin()` does: minimum and maximum are by default the ones of the set of values `x` but can also be set using `mn` and `mx` in which case values below `mn` will be set to `mn` (and thus to 1 in the output) and values above `mx` will be set to `mx` (and thus to 1 in the output). Finally, the number of colors to be used used in is given by `n`.

Value

A numeric vector

Examples

```
x <- stats::rpois(20, 20)
scaleWithin(x, 20, 10, 30)
scaleWithin(matrix(x, 10, 2), mn = 20, mx = 30)
```

---

seqRg *Generate a regular sequence based on the range of a vector*

Description

Generate a regular sequence based on the range of a vector

Usage

```
seqRg(x, n, offset = 0, prop = TRUE)

seqCol(df)

seqRow(df)
```

Arguments

`x`	a vector.
`n`	the number of values in the output sequence.
`offset`	extend or reduce the range .
`prop`	a logical. If `TRUE`, then `offset` is a proportion of the range, otherwise `offset` is used as is.
`df`	a data frame or a matrix.

Functions

- `seqCol()`: a regular sequence based on the number of columns in a data frame.
- `seqRow()`: a regular sequence based on the number of rows in a data frame.

Examples

```
seqRg(runif(10), 10)
seqRg(1:10, 10, .5)
seqCol(matrix(1, 3, 2))
```

`setColClass` *Set class of data frames columns*

Description

This function is designed to ease the changes of column's class of a given data frame.

Usage

```
setColClass(x, colid, cls)
```

Arguments

`x`	a data frame or an R object to be coerced into a data frame.
`colid`	the identity of columns for which class are to be changed.
`cls`	a character vector containing the classes' names to be used in the same order as `colid`. By default, `cls` is repeated until its size equals `colid`'s size.

Value

A data frame whose columns have the desired classes.

Examples

```
df1 <- matrix(signif(runif(20),4), ncol=2)
df2 <- setColClass(df1, 2, 'character')
str(df1)
str(df2)
```

`signifSymbols` *A simple function to associated p-values with symbols*

Description

The `signifSymbols` function takes one vector of p-values and returns a vector of symbols that correspond to thresholds that can be set. Default thresholds values and symbols are the most common ones.

Usage

```
signifSymbols(
  pvalue,
  thresholds = c(0.1, 0.05, 0.01, 0.001),
  symbols = c(".", "*", "**", "***"),
  notsignif = "n.s."
)
```

Arguments

pvalue	a p-value for which a symbol is requested.
thresholds	the threshold values that define category to which symbols are assigned.
symbols	list of symbols.
notsignif	symbols for non significant p-value.

Examples

```
signifSymbols(.012)
signifSymbols(.008)
lapply(c(.2, .08, .04, .008, 0.0001), signifSymbols)
```

---

squaretize *Makes a data frame or a matrix square*

Description

Add missing rows or columns based on names and Duplicates rows and columns of a given a data frame.

Usage

```
squaretize(x, fill = 0, reorder = TRUE)
```

Arguments

x	an object to be coerced into matrix.
fill	character string used to fill the additional rows and/or columns.
reorder	a logical. Should names must be used to order rows and columns?

Value

A square matrix.

Examples

```
mat <- matrix(1:12, 3, 4)
mat2 <- squaretize(mat)
matb <- matrix(1:12, 4, 3)
colnames(matb) <- LETTERS[1:3]
mat2b <- squaretize(matb)
```

`stopwatch` *Timer and Stopwatch*

Description

Timer and Stopwatch

Usage

```
stopwatch(max_time = 1000, pause = 0.01, digits = 5)

timer(time, pause = 0.01, digits = 5)
```

Arguments

`max_time`	time in seconds after which the stopwatch is forced to stop, use `Inf` to run it endlessly.
`pause`	time duration of the pause between 2 time increments displayed (it should better be small).
`digits`	number of digits displayed.
`time`	initial time (in seconds) for the timer.

Functions

- `timer()`: Timer

Examples

```
## Not run:
stopwatch(10)
stopwatch(20)

## End(Not run)
```

tblDown *Write data frames in a document*

Description

Write a data frame or a list of data frames in a markdown document converted #' in various format using pandoc.

Usage

```
tblDown(
  x,
  output_file = "./tables.docx",
  section = NULL,
  caption = NULL,
  title = NULL,
  row.names = FALSE,
  ...
)
```

Arguments

x	a data frame or a list of data frames.
output_file	path to the output file. Its extension will be used by pandoc to correctly render the final document in the write format.
section	a vector of character strings used as section titles (optional).
caption	a vector of character strings used as captions (optional).
title	a character string used as a title for the document (optional).
row.names	a logical. Should row names be added? See `knitr::kable()`.
...	further arguments passed to `knitr::kable()`.

Details

This function calls `base::cat()` and `knitr::kable()` to write a Markdown document containing a list of tables that is then converted into the desired format. For `section` and `caption` if the length differ then will be cut off or expanded.

Value

A data frame whose columns have the desired classes.

References

https://pandoc.org/MANUAL.html#tables

Examples

```
## Not run:
data(CO2)
tblDown(list(CO2[1:2, ], CO2[3:6,]), section = "section")
tblDown(list(CO2[1:2, ], CO2[3:6,]), "./tables.pdf")

## End(Not run)
```

`whichIs` *Search values or patterns in a vector*

Description

Search values or patterns in a vector.

Usage

```
whichIs(x, y, isPattern = FALSE, ...)
```

Arguments

`x`	a vector of values or patterns to be searched for.
`y`	a vector where x values are searched for.
`isPattern`	should x be considered as a vector of patterns?
`...`	further arguments to be passed to `base::which()`.

Value

A list of positions. Each element corresponds to the matches for a specific x value.

Examples

```
vec <- LETTERS[1:10]
spl <- sample(vec)
id <- unlist(whichIs(vec, spl))
identical(vec, spl[id])
```

Index