

Package ‘rtemisutils’

February 7, 2026

Version 0.1.0

Title Shared Utilities for Rtemis Packages

Date 2026-02-01

Description Includes a collection of type-checking and text-formatting utilities as well as the rtemis color system. This package is not intended for direct use by users, but rather to be imported by other rtemis packages.

License GPL (>= 3)

URL <https://www.rtemis.org>

BugReports <https://github.com/rtemis-org/rtemisutils/issues>

ByteCompile yes

Depends R (>= 4.1.0)

Imports cli, data.table, grDevices, graphics, htmltools, S7, stats, methods, utils

Suggests car, DBI, duckdb, future, gsubfn, igraph, parallelly, plotly, reactable, reticulate, seqinr, testthat, timeDate

Encoding UTF-8

Config/testthat.edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

NeedsCompilation no

Author E.D. Gennatas [aut, cre, cph] (ORCID:
[<https://orcid.org/0000-0001-9280-3609>](https://orcid.org/0000-0001-9280-3609))

Maintainer E.D. Gennatas <gennatas@gmail.com>

Contents

rtemisutils-package	3
ansi256_to_hex	4
bold	4
check_dependencies	5
check_inherits	6
clean_colnames	6
clean_int	7

clean_names	8
col256	8
col2grayscale	9
col2hex	10
colorgrad	10
color_adjust	13
color_fade	13
color_invertRGB	14
color_mix	14
color_op	15
color_txt_columns	16
ddSci	17
desaturate	18
df_movecolumn	18
drange	19
dt_describe	20
dt_inspect_types	21
dt_keybin_reshape	21
dt_merge	22
dt_names_by_attr	24
dt_nunique_perfeat	24
dt_pctmatch	25
dt_pctmissing	26
dt_set_autotypes	26
dt_set_cleanfactorlevels	27
dt_set_clean_all	28
dt_set_logical2factor	29
export_plotly	30
factor_NA2missing	31
fct_describe	31
filter_order	32
fmt	33
fmt_gradient	34
getnames	35
getnamesandtypes	36
get_loaded_pkg_version	36
get_mode	37
get_output_type	37
get_vars_from_rules	38
graph_node_metrics	39
gray	39
green	40
highlight	40
iflengthy	41
index_col_by_attr	42
init_project_dir	42
inspect_type	43
is_constant	44
is_discrete	44
italic	45
labelify	45
list2csv	46

lotri2edgeList	47
make_path	48
matchcases	48
mgetnames	50
msg	51
msgdatetime	52
msgdone	52
msgstart	53
muted	53
names_by_class	54
orange	55
pcat	55
previewcolor	56
printdf	57
printls	58
qstat	59
recycle	60
repr	60
repr_ls	61
repr_S7name	62
rnormmat	63
rtemis_colors	64
rtpalette	65
rtversion	65
rt_letters	66
rt_reactable	66
rules2medmod	67
runifmat	68
setdiffsym	69
sge_submit	69
show_pad	71
show_range	71
size	72
table_column_attr	72
thin	73
underline	74
uniprot_get	74
uniquevalsperfeat	75
vec2df	76
xtdescribe	76

Index

78

Description

Author(s)

Maintainer: E.D. Gennatas <gennatas@gmail.com> ([ORCID](#)) [copyright holder]

See Also

Useful links:

- <https://www.rtemis.org>
- Report bugs at <https://github.com/rtemis-org/rtemisutils/issues>

`ansi256_to_hex`

Convert ANSI 256 color code to HEX

Description

Convert ANSI 256 color code to HEX

Usage

```
ansi256_to_hex(code)
```

Arguments

`code` Integer: ANSI 256 color code (0-255).

Value

Character: HEX color string.

Author(s)

EDG

`bold`

Make text bold

Description

A `fmt()` convenience wrapper for making text bold.

Usage

```
bold(text, output_type = c("ansi", "html", "plain"))
```

Arguments

`text` Character: Text to make bold

`output_type` Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with bold styling

Author(s)

EDG

check_dependencies **rtemis internal:** *Dependencies check*

Description

Checks if dependencies can be loaded; names missing dependencies if not.

Usage

```
check_dependencies(..., verbosity = 0L)
```

Arguments

- | | |
|-----------|--|
| ... | List or vector of strings defining namespaces to be checked |
| verbosity | Integer: Verbosity level. Note: An error will always be printed if dependencies are missing. Setting this to FALSE stops it from printing "Dependencies check passed". |

Value

Called for side effects. Aborts and prints list of missing dependencies, if any.

Author(s)

EDG

Examples

```
# This will throw an error if "unavailable" is not installed:  
# check_dependencies("unavailable")
```

`check_inherits` *Test class of object*

Description

Test class of object

Usage

```
check_inherits(x, cl, allow_null = TRUE, xname = deparse(substitute(x)))
```

Arguments

- | | |
|-------------------------|---|
| <code>x</code> | Object to check. |
| <code>cl</code> | Character: class to check against. |
| <code>allow_null</code> | Logical: If TRUE, NULL values are allowed and return early. |

Details

Exported as internal function for use by other rtemis packages.

Value

Called for side effects. Throws an error if checks fail.

Author(s)

EDG

`clean_colnames` *Clean column names*

Description

Clean column names by replacing all spaces and punctuation with a single underscore

Usage

```
clean_colnames(x)
```

Arguments

- | | |
|----------------|---|
| <code>x</code> | Character vector or matrix with colnames or any object with names() method. |
|----------------|---|

Value

Character vector.

Author(s)

EDG

Examples

```
clean_colnames(iris)
```

clean_int	<i>Clean integer input</i>
-----------	----------------------------

Description

Clean integer input

Usage

```
clean_int(x, xname = deparse(substitute(x)))
```

Arguments

- | | |
|-------|---|
| x | Double or integer vector to check. |
| xname | Character: Name of x, for error messages. |

Details

The goal is to return an integer vector. If the input is integer, it is returned as is. If the input is numeric, it is coerced to integer only if the numeric values are integers, otherwise an error is thrown.

Value

Integer vector

Author(s)

EDG

Examples

```
## Not run:  
clean_int(6L)  
clean_int(3)  
clean_int(12.1) # Error  
clean_int(c(3, 5, 7))  
clean_int(c(3, 5, 7.01)) # Error  
  
## End(Not run)
```

<code>clean_names</code>	<i>Clean names</i>
--------------------------	--------------------

Description

Clean character vector by replacing all symbols and sequences of symbols with single underscores, ensuring no name begins or ends with a symbol

Usage

```
clean_names(x, prefix_digits = "V_")
```

Arguments

- `x` Character vector.
- `prefix_digits` Character: prefix to add to names beginning with a digit. Set to NA to skip.

Value

Character vector.

Author(s)

EDG

Examples

```
x <- c("Patient ID", "_Date-of-Birth", "SBP (mmHg)")
x
clean_names(x)
```

<code>col256</code>	<i>Apply 256-color formatting</i>
---------------------	-----------------------------------

Description

Apply 256-color formatting

Usage

```
col256(text, col = "79", bg = FALSE, output_type = c("ansi", "html", "plain"))
```

Arguments

- `text` Character: Text to color
- `col` Character or numeric: Color (ANSI 256-color code, hex for HTML)
- `bg` Logical: If TRUE, apply as background color
- `output_type` Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with 256-color styling

Author(s)

EDG

col2grayscale	<i>Color to Grayscale</i>
---------------	---------------------------

Description

Convert a color to grayscale

Usage

```
col2grayscale(x, what = c("color", "decimal"))
```

Arguments

x	Color to convert to grayscale
what	Character: "color" returns a hexadecimal color, "decimal" returns a decimal between 0 and 1

Details

Uses the NTSC grayscale conversion: $0.299 * R + 0.587 * G + 0.114 * B$

Value

Character: color hex code.

Author(s)

EDG

Examples

```
## Not run:  
col2grayscale("red")  
col2grayscale("red", "dec")  
  
## End(Not run)
```

`col2hex`*Convert R color to hexadecimal code***Description**

Convert a color that R understands into the corresponding hexadecimal code

Usage

```
col2hex(color)
```

Arguments

<code>color</code>	Color(s) that R understands
--------------------	-----------------------------

Value

Character vector of hexadecimal codes.

Author(s)

EDG

Examples

```
## Not run:
col2hex(c("gray50", "skyblue"))

## End(Not run)
```

`colorgrad`*Color Gradient***Description**

Create a gradient of colors and optionally a colorbar

Usage

```
colorgrad(
  n = 21,
  colors = NULL,
  space = c("rgb", "Lab"),
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  preview = FALSE,
  colorbar = FALSE,
```

```

cb_n = 21,
cb_mar = c(1, 1, 1, 1),
cb_add = FALSE,
cb_add_mar = c(5, 0, 2, 5),
cb_axis_pos = 1.1,
cb_axis_las = 1,
cb_axis_hadj = 0,
cb_cex = 6,
bar_min = -1,
bar_mid = 0,
bar_max = 1,
cex = 1.2,
filename = NULL,
pdf_width = 3,
pdf_height = 7,
theme = getOption("rt.theme", "light"),
bg = NULL,
col_text = NULL,
plotlycb = FALSE,
plotly_width = 80,
plotly_height = 500,
return_plotly = FALSE,
margins = c(0, 0, 0, 0),
pad = 0,
par_reset = TRUE
)

```

Arguments

n	Integer: How many distinct colors you want. If not odd, converted to n + 1 Defaults to 21
colors	Character: Acts as a shortcut to defining lo, mid, etc for a number of defaults: "french", "penn", "grnblkred",
space	Character: Which colorspace to use. Option: "rgb", or "Lab". Recommendation: If mid is "white" or "black" (default), use "rgb", otherwise "Lab"
lo	Color for low end
lomid	Color for low-mid
mid	Color for middle of the range or "mean", which will result in color_op(c(lo, hi), "mean"). If mid = NA, then only lo and hi are used to create the color gradient.
midhi	Color for middle-high
hi	Color for high end
preview	Logical: Plot the colors horizontally
colorbar	Logical: Create a vertical colorbar
cb_n	Integer: How many steps you would like in the colorbar
cb_mar	Vector, length 4: Colorbar margins. Default: c(1, 1, 1, 1)
cb_add	Logical: If TRUE, colorbar will be added to existing plot
cb_add_mar	Vector: Margins for colorbar (See par("mar"))
cb_axis_pos	Float: Position of axis (See axis("pos"))

<code>cb_axis_las</code>	Integer {0,1,2,3}: Style of axis labels. 0: Always parallel to the axis, 1: Horizontal, 2: Perpendicular, 3: Vertical.
<code>cb_axis_hadj</code>	Float: Adjustment parallel to the reading direction (See <code>par("adj")</code>)
<code>cb_cex</code>	Float: Character expansion factor for colorbar (See <code>par("cex")</code>)
<code>bar_min</code>	Numeric: Lowest value in colorbar
<code>bar_mid</code>	Numeric: Middle value in colorbar
<code>bar_max</code>	Numeric: Max value in colorbar
<code>cex</code>	Float: Character expansion for axis
<code>filename</code>	String (Optional: Path to file to save colorbar)
<code>pdf_width</code>	Float: Width for PDF output.
<code>pdf_height</code>	Float: Height for PDF output.
<code>theme</code>	Theme object.
<code>bg</code>	Color: Background color
<code>col_text</code>	Color: Colorbar text color
<code>plotlycb</code>	Logical: Create colorbar using <code>plotly</code> (instead of base R graphics)
<code>plotly_width</code>	Float: Width for <code>plotly</code> colorbar.
<code>plotly_height</code>	Float: Height for <code>plotly</code> colorbar.
<code>return_plotly</code>	Logical: If TRUE, return <code>plotly</code> object
<code>margins</code>	Vector: <code>plotly</code> margins.
<code>pad</code>	Float: Padding for <code>plotly</code> .
<code>par_reset</code>	Logical: If TRUE (Default), reset <code>par</code> settings after running

Details

It is best to provide an odd number, so that there is always an equal number of colors on either side of the midpoint. For example, if you want a gradient from -1 to 1 or equivalent, an `n = 11`, will give 5 colors on either side of 0, each representing a 20%

`colors` can be defined as a sequence of 3-letter color abbreviations of 2, 3, 4, or 5 colors which will correspond to values: `{"lo", "hi"}`; `{"lo", "mid", "hi"}`; `{"lo", "mid", "midhi", "hi"}`, and `{"lo", "lo-mid", "mid", "midhi", "hi"}`, respectively. For example, try `colorgrad(21, "blugrnblkredyel", colorbar = TRUE)` 3-letter color abbreviations: wht: white; blk: black; red: red; grn: green; blu: blue; yel: yellow; rng: orange; prl: purple

Value

Invisible vector of hexadecimal colors / `plotly` object if `return_plotly = TRUE`

Author(s)

EDG

color_adjust	<i>Adjust HSV Color</i>
--------------	-------------------------

Description

Modify alpha, hue, saturation and value (HSV) of a color

Usage

```
color_adjust(color, alpha = NULL, hue = 0, sat = 0, val = 0)
```

Arguments

color	Input color. Any format that grDevices::col2rgb() recognizes
alpha	Numeric: Scale alpha by this amount. Future: replace with absolute setting
hue	Float: How much hue to add to color
sat	Float: How much saturation to add to color
val	Float: How much to increase value of color by

Value

Adjusted color

Author(s)

EDG

color_fade	<i>Fade color towards target</i>
------------	----------------------------------

Description

Fade color towards target

Usage

```
color_fade(x, to = "#000000", pct = 0.5)
```

Arguments

x	Color source
to	Target color
pct	Numeric (0, 1) fraction of the distance in RGBA space between x and to to move. e.g. .5 gets the mean RGBA value of the two

Value

Color in hex notation

Author(s)

EDG

color_invertRGB	<i>Invert Color in RGB space</i>
-----------------	----------------------------------

Description

Invert Color in RGB space

Usage

color_invertRGB(x)

Arguments

x	Color, vector
---	---------------

Value

Inverted colors using hexadecimal notation #RRGGBBAA

Author(s)

EDG

Examples

```
## Not run:
cols <- c("red", "green", "blue")
previewcolor(cols)
cols |>
  color_invertRGB() |>
  previewcolor()

## End(Not run)
```

color_mix	<i>Create an alternating sequence of graded colors</i>
-----------	--

Description

Create an alternating sequence of graded colors

Usage

color_mix(color, n = 4)

Arguments

color	List: List of two or more elements, each containing two colors. A gradient will be created from the first to the second color of each element
n	Integer: Number of steps in each gradient.

Value

Character vector of color hex codes.

Author(s)

EDG

Examples

```
## Not run:
color <- list(
  blue = c("#82af3", "#000f3a"),
  gray = c("gray10", "gray85")
)
previewcolor(desaturate(color_mix(color, 6), .3))

color <- list(
  blue = c("#82af3", "#57000a"),
  gray = c("gray10", "gray85")
)
previewcolor(desaturate(color_mix(color, 6), .3))

color <- list(
  blue = c("#82af3", "#000f3a"),
  purple = c("#23001f", "#c480c1")
)
previewcolor(desaturate(color_mix(color, 5), .3))

## End(Not run)
```

Description

Invert a color or calculate the mean of two colors in HSV or RGB space. This may be useful in creating colors for plots

Usage

```
color_op(col, fn = c("invert", "mean"), space = c("HSV", "RGB"))
```

Arguments

col	Input color(s)
fn	Character: "invert", "mean": Function to perform
space	Character: "HSV", "RGB": Colorspace to operate in - for averaging only

Details

The average of two colors in RGB space will often pass through gray, which is likely undesirable. Averaging in HSV space, better for most applications.

Value

Color

Author(s)

EDG

<code>color_txt_columns</code>	<i>Color columns of text art</i>
--------------------------------	----------------------------------

Description

This function accepts text input of 1 or more lines and two colors. It will: a) generate a color gradient between the two colors b) apply the gradient to each column of the text, creating a left to right color gradient.

Usage

```
color_txt_columns(
  x,
  color_left,
  color_right,
  output_type = c("ansi", "html", "plain")
)
```

Arguments

<code>x</code>	Character vector of text to colorize.
<code>color_left</code>	Color for the left side of the gradient.
<code>color_right</code>	Color for the right side of the gradient.
<code>output_type</code>	Character: Output type. One of "ansi", "html", "plain". Default = "ansi".

Value

Character vector with color formatting applied to each column.

Author(s)

EDG

ddSci*Format Numbers for Printing*

Description

2 Decimal places, otherwise scientific notation

Usage

```
ddSci(x, decimal_places = 2, hi = 1e+06, as_numeric = FALSE)
```

Arguments

x	Vector of numbers
decimal_places	Integer: Return this many decimal places.
hi	Float: Threshold at or above which scientific notation is used.
as_numeric	Logical: If TRUE, convert to numeric before returning. This will not force all numbers to print 2 decimal places. For example: 1.2035 becomes "1.20" if as_numeric = FALSE, but 1.2 otherwise. This can be helpful if you want to be able to use the output as numbers / not just for printing.

Details

Numbers will be formatted to 2 decimal places, unless this results in 0.00 (e.g. if input was .0032), in which case they will be converted to scientific notation with 2 significant figures. ddSci will return 0.00 if the input is exactly zero. This function can be used to format numbers in plots, on the console, in logs, etc.

Value

Formatted number

Author(s)

EDG

Examples

```
x <- .34876549
ddSci(x)
# "0.35"
x <- .00000000457823
ddSci(x)
# "4.6e-09"
```

desaturate	<i>Pastelify a color (make a color more pastel)</i>
------------	---

Description

Lower a color's saturation by a given percent in the HSV color system

Usage

```
desaturate(x, s = 0.3)
```

Arguments

x	Color, vector: Color(s) to operate on
s	Float: Decrease saturation by this fraction. Default = .3, which means if saturation of given color is 1, it will become .7

Value

List of adjusted colors

Author(s)

EDG

Examples

```
## Not run:
cols <- c("red", "green", "blue")
previewcolor(cols)
cols_d <- desaturate(cols)
previewcolor(cols_d)

## End(Not run)
```

df_movecolumn	<i>Move data frame column</i>
---------------	-------------------------------

Description

Move data frame column

Usage

```
df_movecolumn(x, colname, to = ncol(x))
```

Arguments

x	data.frame.
colname	Character: Name of column you want to move.
to	Integer: Which column position to move the vector to. Default = ncol(x) i.e. the last column.

Value

data.frame

Author(s)

EDG

Examples

```
## Not run:  
ir <- df_movecolumn(iris, colname = "Species", to = 1L)  
  
## End(Not run)
```

drange

Set Dynamic Range

Description

rtemis preproc: Adjusts the dynamic range of a vector or matrix input. By default normalizes to 0-1 range.

Usage

```
drange(x, lo = 0, hi = 1, byCol = TRUE)
```

Arguments

x	Numeric vector or matrix / data frame: Input
lo	Target range minimum. Defaults to 0
hi	Target range maximum. Defaults to 1
byCol	Logical: If TRUE: if x is matrix, drange each column separately

Value

Numeric vector.

Author(s)

EDG

Examples

```
## Not run:  
x <- runif(20, -10, 10)  
x <- drange(x)  
  
## End(Not run)
```

dt_describe *Describe data.table*

Description

Describe data.table

Usage

```
dt_describe(x, verbosity = 1L)
```

Arguments

x	data.table: Input data.table.
verbosity	Integer: If > 0, print output to console.

Value

List with three data.tables: Numeric, Categorical, and Date.

Author(s)

EDG

Examples

```
library(data.table)
origin <- as.POSIXct("2022-01-01 00:00:00", tz = "America/Los_Angeles")
x <- data.table(
  ID = paste0("ID", 1:10),
  V1 = rnorm(10),
  V2 = rnorm(10, 20, 3),
  V1_datetime = as.POSIXct(
    seq(
      1, 1e7,
      length.out = 10
    ),
    origin = origin
  ),
  V2_datetime = as.POSIXct(
    seq(
      1, 1e7,
      length.out = 10
    ),
    origin = origin
  ),
  C1 = sample(c("alpha", "beta", "gamma"), 10, TRUE),
  F1 = factor(sample(c("delta", "epsilon", "zeta"), 10, TRUE))
)
```

dt_inspect_types *Inspect column types*

Description

Will attempt to identify columns that should be numeric but are either character or factor by running [inspect_type](#) on each column.

Usage

```
dt_inspect_types(x, cols = NULL, verbosity = 1L)
```

Arguments

x	data.table: Input data.table.
cols	Character vector: columns to inspect.
verbosity	Integer: Verbosity level.

Value

Character vector.

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(
  id = 8001:8006,
  a = c("3", "5", "undefined", "21", "4", NA),
  b = c("mango", "banana", "tangerine", NA, "apple", "kiwi"),
  c = c(1, 2, 3, 4, 5, 6)
)
dt_inspect_types(x)
```

dt_keybin_reshape *Long to wide key-value reshaping*

Description

Reshape a long format `data.table` using key-value pairs with `data.table::dcast`

Usage

```
dt_keybin_reshape(
  x,
  id_name,
  key_name,
  positive = 1,
  negative = 0,
  xname = NULL,
  verbosity = 1L
)
```

Arguments

<code>x</code>	<code>data.table</code> object.
<code>id_name</code>	Character: Name of column in <code>x</code> that defines the IDs identifying individual rows.
<code>key_name</code>	Character: Name of column in <code>x</code> that holds the key.
<code>positive</code>	Numeric or Character: Used to fill <code>id ~ key</code> combination present in the long format input <code>x</code> .
<code>negative</code>	Numeric or Character: Used to fill <code>id ~ key</code> combination NOT present in the long format input <code>x</code> .
<code>xname</code>	Character: Name of <code>x</code> to be used in messages.
<code>verbosity</code>	Integer: Verbosity level.

Value

`data.table` in wide format.

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(
  ID = rep(1:3, each = 2),
  Dx = c("A", "C", "B", "C", "D", "A")
)
dt_keybin_reshape(x, id_name = "ID", key_name = "Dx")
```

Description

Merge data.tables

Usage

```
dt_merge(
  left,
  right,
  on = NULL,
  left_on = NULL,
  right_on = NULL,
  how = "left",
  left_name = NULL,
  right_name = NULL,
  left_suffix = NULL,
  right_suffix = NULL,
  verbosity = 1L,
  ...
)
```

Arguments

<code>left</code>	<code>data.table</code>
<code>right</code>	<code>data.table</code>
<code>on</code>	Character: Name of column to join on.
<code>left_on</code>	Character: Name of column on left table.
<code>right_on</code>	Character: Name of column on right table.
<code>how</code>	Character: Type of join: "inner", "left", "right", "outer".
<code>left_name</code>	Character: Name of left table.
<code>right_name</code>	Character: Name of right table.
<code>left_suffix</code>	Character: If provided, add this suffix to all left column names, excluding on/left_on.
<code>right_suffix</code>	Character: If provided, add this suffix to all right column names, excluding on/right_on.
<code>verbosity</code>	Integer: Verbosity level.
<code>...</code>	Additional arguments to be passed to <code>data.table::merge</code> .

Value

Merged `data.table`.

Author(s)

EDG

Examples

```
library(data.table)
xleft <- data.table(ID = 1:5, Alpha = letters[1:5])
xright <- data.table(ID = c(3, 4, 5, 6), Beta = LETTERS[3:6])
xlr_inner <- dt_merge(xleft, xright, on = "ID", how = "inner")
```

`dt_names_by_attr` *List column names by attribute*

Description

List column names by attribute

Usage

```
dt_names_by_attr(x, attribute, exact = TRUE, sorted = TRUE)
```

Arguments

<code>x</code>	data.table: Input data.table.
<code>attribute</code>	Character: name of attribute.
<code>exact</code>	Logical: If TRUE, use exact matching.
<code>sorted</code>	Logical: If TRUE, sort the output.

Value

Character vector.

Author(s)

EDG

`dt_nunique_perfeat` *Number of unique values per feature*

Description

Number of unique values per feature

Usage

```
dt_nunique_perfeat(x, excludeNA = FALSE, limit = 20L, verbosity = 1L)
```

Arguments

<code>x</code>	data.table: Input data.table.
<code>excludeNA</code>	Logical: If TRUE, exclude NA values.
<code>limit</code>	Integer: Print up to this many features. Set to -1L to print all.
<code>verbosity</code>	Integer: If > 0, print output to console.

Value

Named integer vector of length `NCOL(x)` with number of unique values per column/feature, invisibly.

Author(s)

EDG

Examples

```
library(data.table)
ir <- as.data.table(iris)
dt_nunique_perfeat(ir)
```

dt_pctmatch*Get N and percent match of values between two columns of two data.tables*

Description

Get N and percent match of values between two columns of two data.tables

Usage

```
dt_pctmatch(x, y, on = NULL, left_on = NULL, right_on = NULL, verbosity = 1L)
```

Arguments

x	data.table: First input data.table.
y	data.table: Second input data.table.
on	Integer or character: column to read in x and y, if it is the same
left_on	Integer or character: column to read in x
right_on	Integer or character: column to read in y
verbosity	Integer: Verbosity level.

Value

list.

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(ID = 1:5, Alpha = letters[1:5])
y <- data.table(ID = c(3, 4, 5, 6), Beta = LETTERS[3:6])
dt_pctmatch(x, y, on = "ID")
```

dt_pctmissing*Get percent of missing values from every column***Description**

Get percent of missing values from every column

Usage

```
dt_pctmissing(x, verbosity = 1L)
```

Arguments

x	data.frame or data.table
verbosity	Integer: Verbosity level.

Value

list

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(a = c(1, 2, NA, 4), b = c(NA, NA, 3, 4), c = c("A", "B", "C", NA))
dt_pctmissing(x)
```

dt_set_autotypes*Set column types automatically***Description**

This function inspects a data.table and attempts to identify columns that should be numeric but have been read in as character, and fixes their type **in-place**. This can happen when one or more fields contain non-numeric characters, for example.

Usage

```
dt_set_autotypes(x, cols = NULL, verbosity = 1L)
```

Arguments

x	data.table: Input data.table. Will be modified in-place , if needed.
cols	Character vector: columns to work on. If not defined, will work on all columns
verbosity	Integer: Verbosity level.

Value

data.table, invisibly.

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(
  id = 8001:8006,
  a = c("3", "5", "undefined", "21", "4", NA),
  b = c("mango", "banana", "tangerine", NA, "apple", "kiwi"),
  c = c(1, 2, 3, 4, 5, 6)
)
str(x)
# ***in-place*** operation means no assignment is needed
dt_set_autotypes(x)
str(x)

# Try excluding column 'a' from autotyping
x <- data.table(
  id = 8001:8006,
  a = c("3", "5", "undefined", "21", "4", NA),
  b = c("mango", "banana", "tangerine", NA, "apple", "kiwi"),
  c = c(1, 2, 3, 4, 5, 6)
)
str(x)
# exclude column 'a' from autotyping
dt_set_autotypes(x, cols = setdiff(names(x), "a"))
str(x)
```

dt_set_cleanfactorlevels

*Clean factor levels of data.table **in-place***

Description

Finds all factors in a data.table and cleans factor levels to include only underscore symbols

Usage

```
dt_set_cleanfactorlevels(x, prefix_digits = NA)
```

Arguments

<code>x</code>	data.table: Input data.table. Will be modified in-place .
<code>prefix_digits</code>	Character: If not NA, add this prefix to all factor levels that are numbers

Value

Nothing, modifies `x` **in-place**.

Author(s)

EDG

Examples

```
library(data.table)
x <- as.data.table(iris)
levels(x[["Species"]]) <- c("setosa:iris", "versicolor$iris", "virginica iris")
levels(x[["Species"]])
dt_set_cleanfactorlevels(x)
levels(x[["Species"]])
```

dt_set_clean_all*Clean column names and factor levels **in-place*****Description**Clean column names and factor levels **in-place****Usage**`dt_set_clean_all(x, prefix_digits = NA)`**Arguments**

x data.table: Input data.table. Will be modified **in-place**, if needed.
prefix_digits Character: prefix to add to names beginning with a digit. Set to NA to skip

ValueNothing, modifies **x in-place**.**Author(s)**

EDG

Examples

```
library(data.table)
x <- as.data.table(iris)
levels(x[["Species"]]) <- c("setosa:iris", "versicolor$iris", "virginica iris")
names(x)
levels(x[["Species"]])
# ***in-place*** operation means no assignment is needed
dt_set_clean_all(x)
names(x)
levels(x[["Species"]])
```

`dt_set_logical2factor` *Convert data.table logical columns to factors*

Description

Convert data.table logical columns to factors with custom labels **in-place**

Usage

```
dt_set_logical2factor(
  x,
  cols = NULL,
  labels = c("False", "True"),
  maintain_attributes = TRUE,
  fillNA = NULL
)
```

Arguments

<code>x</code>	data.table: Input data.table. Will be modified in-place .
<code>cols</code>	Integer or character: columns to convert, if NULL, operates on all logical columns
<code>labels</code>	Character: labels for factor levels
<code>maintain_attributes</code>	Logical: If TRUE, maintain column attributes
<code>fillNA</code>	Character: If not NULL, fill NA values with this constant

Value

data.table, invisibly.

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(a = 1:5, b = c(TRUE, FALSE, FALSE, FALSE, TRUE))
x
dt_set_logical2factor(x)
x
z <- data.table(
  alpha = 1:5,
  beta = c(TRUE, FALSE, TRUE, NA, TRUE),
  gamma = c(FALSE, FALSE, TRUE, FALSE, NA)
)
# You can use fillNA to fill NA values with a constant
dt_set_logical2factor(z, cols = "beta", labels = c("No", "Yes"), fillNA = "No")
z
w <- data.table(mango = 1:5, banana = c(FALSE, FALSE, TRUE, TRUE, FALSE))
w
dt_set_logical2factor(w, cols = 2, labels = c("Ugh", "Huh"))
```

```
w
# Column attributes are maintained by default:
z <- data.table(
  alpha = 1:5,
  beta = c(TRUE, FALSE, TRUE, NA, TRUE),
  gamma = c(FALSE, FALSE, TRUE, FALSE, NA)
)
for (i in seq_along(z)) setattr(z[[i]], "source", "Guava")
str(z)
dt_set_logical2factor(z, cols = "beta", labels = c("No", "Yes"))
str(z)
```

export_plotly *Export plotly plot to file*

Description

Export plotly plot to file

Usage

```
export_plotly(
  x,
  filename,
  width = 600,
  height = 600,
  scale = 1,
  import_kaleido = TRUE,
  verbosity = 1L
)
```

Arguments

<code>x</code>	plotly object.
<code>filename</code>	Character: Filename to save the plot to.
<code>width</code>	Numeric: Width of the exported image in pixels.
<code>height</code>	Numeric: Height of the exported image in pixels.
<code>scale</code>	Numeric: Scale factor for the exported image.
<code>import_kaleido</code>	Logical: If TRUE, attempts to import kaleido for exporting plotly plots.
<code>verbosity</code>	Integer: Verbosity level.

Author(s)

EDG

factor_NA2missing *Factor NA to "missing" level*

Description

Set NA values of a factor vector to a new level indicating missingness

Usage

```
factor_NA2missing(x, na_level_name = "missing")
```

Arguments

x	Factor.
na_level_name	Character: Name of new level to create that will be assigned to all current NA values in x.

Value

factor.

Author(s)

EDG

Examples

```
x <- factor(sample(letters[1:3], 100, TRUE))
x[sample(1:100, 10)] <- NA
xm <- factor_NA2missing(x)
```

fct_describe *Describe factor*

Description

Outputs a single character with names and counts of each level of the input factor.

Usage

```
fct_describe(x, max_n = 5, return_ordered = TRUE)
```

Arguments

x	factor.
max_n	Integer: Return counts for up to this many levels.
return_ordered	Logical: If TRUE, return levels ordered by count, otherwise return in level order.

Value

Character with level counts.

Author(s)

EDG

Examples

```
## Not run:
# Small number of levels
fct_describe(iris[["Species"]])

# Large number of levels: show top n by count
x <- factor(sample(letters, 1000, TRUE))
fct_describe(x)
fct_describe(x, 3)

## End(Not run)
```

filter_order

Filter order

Description

Filter order

Usage

```
filter_order(x, idl, decreasing = FALSE)
```

Arguments

x	Input vector
idl	Logical vector: Index of elements to filter
decreasing	Logical: If TRUE, sort in descending order

Author(s)

EDG

Examples

```
## Not run:
x <- rnorm(10)
x
x$filter_order(x, x < 0)

## End(Not run)
```

fmt	<i>Text formatting</i>
-----	------------------------

Description

Formats text with specified color, styles, and background using ANSI escape codes or HTML, with support for plain text output.

Usage

```
fmt(  
  x,  
  col = NULL,  
  bold = FALSE,  
  italic = FALSE,  
  underline = FALSE,  
  thin = FALSE,  
  muted = FALSE,  
  bg = NULL,  
  pad = 0L,  
  output_type = c("ansi", "html", "plain")  
)
```

Arguments

x	Character: Text to format.
col	Character: Color (hex code, named color, or NULL for no color).
bold	Logical: If TRUE, make text bold.
italic	Logical: If TRUE, make text italic.
underline	Logical: If TRUE, underline text.
thin	Logical: If TRUE, make text thin/light.
muted	Logical: If TRUE, make text muted/dimmed.
bg	Character: Background color (hex code, named color, or NULL).
pad	Integer: Number of spaces to pad before text.
output_type	Character: Output type ("ansi", "html", "plain").

Details

This function combines multiple formatting options into a single call, making it more efficient than nested function calls. It generates optimized ANSI escape sequences and clean HTML output.

Value

Character: Formatted text with specified styling.

Author(s)

EDG

Examples

```
# Simple color
fmt("Hello", col = "red")

# Bold red text
fmt("Error", col = "red", bold = TRUE)

# Multiple styles
fmt("Warning", col = "yellow", bold = TRUE, italic = TRUE)

# With background
fmt("Highlight", col = "white", bg = "blue", bold = TRUE)
```

fmt_gradient

Gradient text

Description

Gradient text

Usage

```
fmt_gradient(x, colors, bold = FALSE, output_type = c("ansi", "html", "plain"))
```

Arguments

<code>x</code>	Character: Text to colorize.
<code>colors</code>	Character vector: Colors to use for the gradient.
<code>bold</code>	Logical: If TRUE, make text bold.
<code>output_type</code>	Character: Output type ("ansi", "html", "plain").

Value

Character: Text with gradient color applied.

Author(s)

EDG

getnames	<i>Get names by string matching or class</i>
----------	--

Description

Get names by string matching or class

Usage

```
getnames(  
  x,  
  pattern = NULL,  
  starts_with = NULL,  
  ends_with = NULL,  
  ignore_case = TRUE  
)  
  
getfactornames(x)  
  
getnumericnames(x)  
  
getlogicalnames(x)  
  
getcharacternames(x)  
  
getdatenames(x)
```

Arguments

x	object with names() method.
pattern	Character: pattern to match anywhere in names of x.
starts_with	Character: pattern to match in the beginning of names of x.
ends_with	Character: pattern to match at the end of names of x.
ignore_case	Logical: If TRUE, well, ignore case.

Details

For getnames() only: pattern, starts_with, and ends_with are applied sequentially. If more than one is provided, the result will be the intersection of all matches.

Value

Character vector of matched names.

Author(s)

EDG

Examples

```
getnames(iris, starts_with = "Sepal")  
getnames(iris, ends_with = "Width")
```

`getnamesandtypes` *Get data.frame names and types*

Description

Get data.frame names and types

Usage

```
getnamesandtypes(x)
```

Arguments

`x` data.frame / data.table or similar

Value

character vector of column names with attribute "type" holding the class of each column

`get_loaded_pkg_version`
 Get version of all loaded packages (namespaces)

Description

Get version of all loaded packages (namespaces)

Usage

```
get_loaded_pkg_version()
```

Value

Data frame with columns "Package_Name" and "Version".

Author(s)

EDG

Examples

```
get_loaded_pkg_version()
```

<code>get_mode</code>	<i>Get the mode of a factor or integer</i>
-----------------------	--

Description

Returns the mode of a factor or integer

Usage

```
get_mode(x, na.rm = TRUE, getlast = TRUE, retain_class = TRUE)
```

Arguments

<code>x</code>	Vector, factor or integer: Input data.
<code>na.rm</code>	Logical: If TRUE, exclude NAs (using <code>na.exclude(x)</code>).
<code>getlast</code>	Logical: If TRUE, get the last value in case of ties.
<code>retain_class</code>	Logical: If TRUE, output is always same class as input.

Value

The mode of `x`

Author(s)

EDG

Examples

```
x <- c(9, 3, 4, 4, 0, 2, 2, NA)
get_mode(x)
x <- c(9, 3, 2, 2, 0, 4, 4, NA)
get_mode(x)
get_mode(x, getlast = FALSE)
```

<code>get_output_type</code>	<i>Get output type</i>
------------------------------	------------------------

Description

Get output type for printing text.

Usage

```
get_output_type(output_type = c("ansi", "html", "plain"), filename = NULL)
```

Arguments

<code>output_type</code>	Character vector of output types.
<code>filename</code>	Optional Character: Filename for output.

Details

Exported as internal function for use by other itemis packages.

Value

Character with selected output type.

Author(s)

EDG

Examples

```
get_output_type()
```

`get_vars_from_rules` *Extract variable names from rules*

Description

Extract variable names from rules

Usage

```
get_vars_from_rules(rules, unique = FALSE)
```

Arguments

<code>rules</code>	Character vector: Rules.
<code>unique</code>	Logical: If TRUE, return only unique variables.

Value

Character vector: Variable names.

Author(s)

EDG

graph_node_metrics *Node-wise (i.e. vertex-wise) graph metrics*

Description

Node-wise (i.e. vertex-wise) graph metrics

Usage

```
graph_node_metrics(x, verbosity = 1L)
```

Arguments

x	igraph network.
verbosity	Integer: Verbosity level.

Value

`data.frame`.

Author(s)

EDG

Examples

```
## Not run:  
datcor <- cor(rnormmmat(20, 20, seed = 2021))  
datcor[sample(seq(datcor), 250)] <- 0  
x <- igraph:::graph_from_adjacency_matrix(  
  adjmatrix = datcor,  
  mode = "lower",  
  weighted = TRUE,  
  diag = FALSE  
)  
  
graph_node_metrics(x)  
  
## End(Not run)
```

gray *Gray text*

Description

A `fmt()` convenience wrapper for making text gray.

Usage

```
gray(x, output_type = c("ansi", "html", "plain"))
```

Arguments

x	Character: Text to format
output_type	Character: Output type ("ansi", "html", "plain")

Details

Can be useful in contexts where muted is not supported.

Value

Character: Formatted text with gray styling

Author(s)

EDG

green	<i>Make text green</i>
-------	------------------------

Description

Make text green

Usage

```
green(..., bold = FALSE)
```

Arguments

...	Character: Text to colorize.
bold	Logical: If TRUE, make text bold.

Author(s)

EDG

highlight	<i>Highlight text</i>
-----------	-----------------------

Description

A `fmt()` convenience wrapper for highlighting text.

Usage

```
highlight(x, pad = 0L, output_type = c("ansi", "html", "plain"))
```

Arguments

x	Character: Text to highlight.
pad	Integer: Number of spaces to pad before text.
output_type	Character: Output type ("ansi", "html", "plain").

Value

Character: Formatted text with highlight.

Author(s)

EDG

iflengthy*Return object if it has length > 0*

Description

Returns the input object if it has length > 0, else NULL

Usage

iflengthy(x)

Arguments

x	Object
---	--------

Value

x if length(x) > 0, else NULL

Author(s)

EDG

Examples

```
x <- 2:4
iflengthy(x)
y <- list()
iflengthy(y)
```

`index_col_by_attr` *Index columns by attribute name & value*

Description

Index columns by attribute name & value

Usage

```
index_col_by_attr(x, name, value)
```

Arguments

<code>x</code>	data.frame or similar.
<code>name</code>	Character: Name of attribute.
<code>value</code>	Character: Value of attribute.

Value

Integer vector.

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(
  id = 1:5,
  sbp = rnorm(5, 120, 15),
  dbp = rnorm(5, 80, 10),
  paO2 = rnorm(5, 90, 10),
  paCO2 = rnorm(5, 40, 5)
)
setattr(x[["sbp"]], "source", "outpatient")
setattr(x[["dbp"]], "source", "outpatient")
setattr(x[["paO2"]], "source", "icu")
setattr(x[["paCO2"]], "source", "icu")
index_col_by_attr(x, "source", "icu")
```

`init_project_dir` *Initialize Project Directory*

Description

Initializes Directory Structure: "R", "Data", "Results"

Usage

```
init_project_dir(verbosity = 1L)
```

Arguments

verbosity Integer: Verbosity level.

Value

Character: the working directory path, invisibly.

Author(s)

EDG

inspect_type *Inspect character and factor vector*

Description

Checks character or factor vector to determine whether it might be best to convert to numeric.

Usage

```
inspect_type(x, xname = NULL, verbosity = 1L, thresh = 0.5, na.omit = TRUE)
```

Arguments

x Character or factor vector.
xname Character: Name of input vector x.
verbosity Integer: Verbosity level.
thresh Numeric: Threshold for determining whether to convert to numeric.
na.omit Logical: If TRUE, remove NA values before checking.

Details

All data can be represented as a character string. A numeric variable may be read as a character variable if there are non-numeric characters in the data. It is important to be able to automatically detect such variables and convert them, which would mean introducing NA values.

Value

Character.

Author(s)

EDG

Examples

```
x <- c("3", "5", "undefined", "21", "4", NA)
inspect_type(x)
z <- c("mango", "banana", "tangerine", NA)
inspect_type(z)
```

is_constant *Check if vector is constant*

Description

Check if vector is constant

Usage

```
is_constant(x, skip_missing = FALSE)
```

Arguments

<code>x</code>	Vector: Input
<code>skip_missing</code>	Logical: If TRUE, skip NA values before test

Value

Logical.

Author(s)

EDG

Examples

```
## Not run:
x <- rep(9, 1000000)
is_constant(x)
x[10] <- NA
is_constant(x)
is_constant(x, skip_missing = TRUE)

## End(Not run)
```

is_discrete *Check if variable is discrete (factor or integer)*

Description

Check if variable is discrete (factor or integer)

Usage

```
is_discrete(x)
```

Arguments

<code>x</code>	Input
----------------	-------

Value

Logical.

Author(s)

EDG

italic*Make text italic***Description**

A `fmt()` convenience wrapper for making text italic.

Usage

```
italic(text, output_type = c("ansi", "html", "plain"))
```

Arguments

<code>text</code>	Character: Text to make italic
<code>output_type</code>	Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with italic styling

Author(s)

EDG

labelify*Format text for label printing***Description**

Format text for label printing

Usage

```
labelify(  
  x,  
  underscores_to_spaces = TRUE,  
  dotsToSpaces = TRUE,  
  toLower = FALSE,  
  toTitleCase = TRUE,  
  capitalize_strings = c("id"),  
  stringsToSpaces = c("\\$", "`")  
)
```

Arguments

x	Character: Input
underscores_to_spaces	Logical: If TRUE, convert underscores to spaces.
dotsToSpaces	Logical: If TRUE, convert dots to spaces.
toLower	Logical: If TRUE, convert to lowercase (precedes toTitleCase). Default = FALSE (Good for getting all-caps words converted to title case, bad for abbreviations you want to keep all-caps)
toTitleCase	Logical: If TRUE, convert to Title Case. Default = TRUE (This does not change all-caps words, set toLower to TRUE if desired)
capitalize_strings	Character, vector: Always capitalize these strings, if present. Default = "id"
stringsToSpaces	Character, vector: Replace these strings with spaces. Escape as needed for gsub. Default = "\\\\$\", which formats common input of the type data.frame\$variable

Value

Character vector.

Author(s)

EDG

Examples

```
x <- c("county_name", "total.cost$", "age", "weight.kg")
labelify(x)
```

list2csv

Write list elements to CSV files

Description

Write list elements to CSV files

Usage

```
list2csv(x, outdir)
```

Arguments

x	List containing R objects to be written to CSV (e.g. data.frames, matrices, etc.)
outdir	Character: Path to output directory

Value

Nothing, writes CSV files to outdir.

Author(s)

EDG

Examples

```
## Not run:
x <- list(
  iris = iris,
  iris_normalized = as.data.frame(scale(iris[, -5]))
)
outdir <- "./exports"
list2csv(x, outdir)

## End(Not run)
```

lotri2edgeList

*Connectivity Matrix to Edge List***Description**

Turn the lower triangle of a connectivity matrix (e.g. correlation matrix or similar) to an edge list of the form: Source, Target, Weight

Usage

```
lotri2edgeList(A, filename = NULL, verbosity = 1L)
```

Arguments

A	Square matrix
filename	Character: Path for csv file. Defaults to "commat2edgelist.csv"
verbosity	Integer: Verbosity level.

Details

The output can be read, for example, into gephi

Value

Data frame with columns: NodeA, NodeB, Weight

Author(s)

EDG

Examples

```
A <- matrix(rnorm(100), nrow = 10)
A[lower.tri(A)] <- t(A)[lower.tri(A)]
diag(A) <- 1
edgelist <- lotri2edgeList(A, filename = NULL)
head(edgelist)
```

make_path	<i>Expand, normalize, concatenate, clean path</i>
-----------	---

Description

Expand, normalize, concatenate, clean path

Usage

```
make_path(..., expand_path = TRUE)
```

Arguments

...	Character: Parts of path to concatenate.
expand_path	Logical: If TRUE, expand concatenated path using path.expand .

Value

Character: Path.

Author(s)

EDG

matchcases	<i>Match cases by covariates</i>
------------	----------------------------------

Description

Find one or more cases from a pool data.frame that match cases in a target data.frame. Match exactly and/or by distance (sum of squared distances).

Usage

```
matchcases(
  target,
  pool,
  n_matches = 1,
  target_id = NULL,
  pool_id = NULL,
  exactmatch_factors = TRUE,
  exactmatch_cols = NULL,
  distmatch_cols = NULL,
  norepeats = TRUE,
  ignore_na = FALSE,
  verbosity = 1L
)
```

Arguments

target	data.frame you are matching against.
pool	data.frame you are looking for matches from.
n_matches	Integer: Number of matches to return.
target_id	Character: Column name in target that holds unique cases IDs. Default = NULL, in which case integer case numbers will be used.
pool_id	Character: Same as target_id for pool.
exactmatch_factors	Logical: If TRUE, selected cases will have to exactly match factors available in target.
exactmatch_cols	Character: Names of columns that should be matched exactly.
distmatch_cols	Character: Names of columns that should be distance-matched.
norepeats	Logical: If TRUE, cases in pool can only be chosen once.
ignore_na	Logical: If TRUE, ignore NA values during exact matching.
verbosity	Integer: Verbosity level.

Value

data.frame

Author(s)

EDG

Examples

```
## Not run:
set.seed(2021)
cases <- data.frame(
  PID = paste0("PID", seq(4)),
  Sex = factor(c(1, 1, 0, 0)),
  Handedness = factor(c(1, 1, 0, 1)),
  Age = c(21, 27, 39, 24),
  Var = c(.7, .8, .9, .6),
  Varx = rnorm(4)
)
controls <- data.frame(
  CID = paste0("CID", seq(50)),
  Sex = factor(sample(c(0, 1), 50, TRUE)),
  Handedness = factor(sample(c(0, 1), 50, TRUE, c(.1, .9))),
  Age = sample(16:42, 50, TRUE),
  Var = rnorm(50),
  Vary = rnorm(50)
)

mc <- matchcases(cases, controls, 2, "PID", "CID")

## End(Not run)
```

mgetnames*Get names by string matching multiple patterns***Description**

Get names by string matching multiple patterns

Usage

```
mgetnames(
  x,
  pattern = NULL,
  starts_with = NULL,
  ends_with = NULL,
  ignore_case = TRUE,
  return_index = FALSE
)
```

Arguments

<code>x</code>	Character vector or object with <code>names()</code> method.
<code>pattern</code>	Character vector: pattern(s) to match anywhere in names of <code>x</code> .
<code>starts_with</code>	Character: pattern to match in the beginning of names of <code>x</code> .
<code>ends_with</code>	Character: pattern to match at the end of names of <code>x</code> .
<code>ignore_case</code>	Logical: If TRUE, well, ignore case.
<code>return_index</code>	Logical: If TRUE, return integer index of matches instead of names.

Details

`pattern`, `starts_with`, and `ends_with` are applied and the union of all matches is returned.
`pattern` can be a character vector of multiple patterns to match.

Value

Character vector of matched names or integer index.

Author(s)

EDG

Examples

```
mgetnames(iris, pattern = c("Sepal", "Petal"))
mgetnames(iris, starts_with = "Sepal")
mgetnames(iris, ends_with = "Width")
```

msg *Message with provenance*

Description

Print message to output with a prefix including date and time, and calling function or full call stack

Usage

```
msg(  
  ...,  
  date = TRUE,  
  caller = NULL,  
  call_depth = 1L,  
  caller_id = 1L,  
  newline_pre = FALSE,  
  newline = TRUE,  
  format_fn = plain,  
  sep = " "  
)  
  
msg0(  
  ...,  
  caller = NULL,  
  call_depth = 1,  
  caller_id = 1,  
  newline_pre = FALSE,  
  newline = TRUE,  
  format_fn = plain,  
  sep = ""  
)
```

Arguments

...	Message to print
date	Logical: if TRUE, include date and time in the prefix
caller	Character: Name of calling function
call_depth	Integer: Print the system call path of this depth.
caller_id	Integer: Which function in the call stack to print
newline_pre	Logical: If TRUE begin with a new line.
newline	Logical: If TRUE end with a new line.
format_fn	Function: Formatting function to use on the message text.
sep	Character: Use to separate objects in ...

Details

If `msg` is called directly from the console, it will print `[interactive>]` in place of the call stack.
`msg0`, similar to `paste0`, is `msg(..., sep = "")`

Value

Invisibly: List with call, message, and date

Author(s)

EDG

Examples

```
msg("Hello, world!")
x <- 42L
msg0("The answer is what you think it is (", x, ".")
```

`msgdatetime`

Message datetime()

Description

Message datetime()

Usage

```
msgdatetime(datetime_format = "%Y-%m-%d %H:%M:%S")
```

Arguments

`datetime_format`

Character: Format for the date and time.

Value

Character: Formatted date and time.

Author(s)

EDG

`msgdone`

msgdone

Description

`msgdone`

Usage

```
msgdone(caller = NULL, call_depth = 1, caller_id = 1, sep = " ")
```

Arguments

caller	Character: Name of calling function
call_depth	Integer: Print the system call path of this depth.
caller_id	Integer: Which function in the call stack to print
sep	Character: Use to separate objects in . . .

Author(s)

EDG

msgstart *msgstart*

Description

msgstart

Usage

```
msgstart(..., newline_pre = FALSE, sep = "")
```

Arguments

...	Message to print
newline_pre	Logical: If TRUE begin with a new line.
sep	Character: Use to separate objects in . . .

Author(s)

EDG

muted *Muted text*

Description

A `fmt()` convenience wrapper for making text muted.

Usage

```
muted(x, output_type = c("ansi", "html", "plain"))
```

Arguments

x	Character: Text to format
output_type	Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with muted styling

Author(s)

EDG

`names_by_class` *List column names by class*

Description

List column names by class

Usage

```
names_by_class(x, sorted = TRUE, item_format = highlight, maxlen = 24)
```

Arguments

- | | |
|--------------------------|---|
| <code>x</code> | data.frame or similar. |
| <code>sorted</code> | Logical: If TRUE, sort the output |
| <code>item_format</code> | Function: Function to format each item |
| <code>maxlen</code> | Integer: Maximum number of items to print |

Value

NULL, invisibly.

Author(s)

EDG

Examples

```
names_by_class(iris)
```

orange	<i>Make text orange</i>
--------	-------------------------

Description

Make text orange

Usage

```
orange(..., bold = FALSE)
```

Arguments

...	Character: Text to colorize.
bold	Logical: If TRUE, make text bold.

Author(s)

EDG

pcat	<i>Pad-cat</i>
------	----------------

Description

Pad and concatenate two strings, with optional newline.

Usage

```
pcat(left, right, pad = 17, newline = TRUE)
```

Arguments

left	Character: Left string to pad and print.
right	Character: Right string to print after left.
pad	Integer: Total width to pad the left string to.
newline	Logical: If TRUE, print a newline after the right string.

Author(s)

EDG

Examples

```
## Not run:  
{  
  msg("Hello")  
  pcat("super", "wow")  
  pcat(NULL, "oooo")  
}  
  
## End(Not run)
```

previewcolor	<i>Preview color</i>
--------------	----------------------

Description

Preview one or multiple colors using little rhombi with their little labels up top

Usage

```
previewcolor(
  x,
  main = NULL,
  bg = "#333333",
  main_col = "#b3b3b3",
  main_x = 0.7,
  main_y = 0.2,
  main_adj = 0,
  main_cex = 0.9,
  main_font = 2,
  width = NULL,
  xlim = NULL,
  ylim = c(0, 2.2),
  asp = 1,
  labels_y = 1.55,
  label_cex = NULL,
  mar = c(0, 0, 0, 1),
  par_reset = TRUE,
  filename = NULL,
  pdf_width = 8,
  pdf_height = 2.5
)
```

Arguments

<code>x</code>	Color, vector: One or more colors that R understands
<code>main</code>	Character: Title. Default = <code>NULL</code> , which results in <code>deparse(substitute(x))</code>
<code>bg</code>	Background color.
<code>main_col</code>	Color: Title color
<code>main_x</code>	Float: x coordinate for <code>main</code> .
<code>main_y</code>	Float: y coordinate for <code>main</code> .
<code>main_adj</code>	Float: adj argument to <code>mtext</code> for <code>main</code> .
<code>main_cex</code>	Float: character expansion factor for <code>main</code> .
<code>main_font</code>	Integer, 1 or 2: Weight of <code>main</code> 1: regular, 2: bold.
<code>width</code>	Float: Plot width. Default = <code>NULL</code> , i.e. set automatically
<code>xlim</code>	Vector, length 2: x-axis limits. Default = <code>NULL</code> , i.e. set automatically
<code>ylim</code>	Vector, length 2: y-axis limits.
<code>asp</code>	Float: Plot aspect ratio.

labels_y	Float: y coord for labels. Default = 1.55 (rhombi are fixed and range y .5 - 1.5)
label_cex	Float: Character expansion for labels. Default = NULL, and is calculated automatically based on length of x
mar	Numeric vector, length 4: margin size.
par_reset	Logical: If TRUE, reset par settings on exit.
filename	Character: Path to save plot as PDF.
pdf_width	Numeric: Width of PDF in inches.
pdf_height	Numeric: Height of PDF in inches.

Value

Nothing, prints plot.

Author(s)

EDG

Examples

```
## Not run:
colors <- colorgradient_x(seq(-5, 5))
previewcolor(colors)

## End(Not run)
```

printfdf

Print data frame

Description

Pretty print a data frame

Usage

```
printfdf(
  x,
  pad = 0,
  spacing = 1,
  ddSci_dp = NULL,
  transpose = FALSE,
  justify = "right",
  colnames = TRUE,
  rownames = TRUE,
  column_fmt = highlight,
  row_fmt = gray,
  newline_pre = FALSE,
  newline = FALSE
)
```

Arguments

<code>x</code>	data frame
<code>pad</code>	Integer: Pad output with this many spaces.
<code>spacing</code>	Integer: Number of spaces between columns.
<code>ddSci_dp</code>	Integer: Number of decimal places to print using <code>ddSci</code> . Default = NULL for no formatting.
<code>transpose</code>	Logical: If TRUE, transpose <code>x</code> before printing.
<code>justify</code>	Character: "right", "left".
<code>colnames</code>	Logical: If TRUE, print column names.
<code>rownames</code>	Logical: If TRUE, print row names.
<code>column_fmt</code>	Formatting fn for printing column names.
<code>row_fmt</code>	Formatting fn for printing row names.
<code>newline_pre</code>	Logical: If TRUE, print a new line before printing data frame.
<code>newline</code>	Logical: If TRUE, print a new line after printing data frame.

Details

By design, numbers will not be justified, but using `ddSci_dp` will convert to characters, which will be justified. This is intentional for internal use.

Author(s)

EDG

`printls`

Pretty print list

Description

Pretty print a list (or data frame) recursively

Usage

```
printls(
  x,
  prefix = "",
  pad = 2L,
  item_format = bold,
  maxlen = 4L,
  center_title = TRUE,
  title = NULL,
  title_newline = TRUE,
  newline_pre = FALSE,
  format_fn_rhs = ddSci,
  print_class = TRUE,
  abbrev_class_n = 3L,
  print_df = FALSE,
  print_S4 = FALSE,
  limit = 12L
)
```

Arguments

x	list or object that will be converted to a list.
prefix	Character: Optional prefix for names.
pad	Integer: Pad output with this many spaces.
item_format	Formatting function for list item names.
maxlength	Integer: Maximum length of items to show using headdot() before truncating with ellipsis.
center_title	Logical: If TRUE, autopad title for centering, if present.
title	Character: Optional title to print before list.
title_newline	Logical: If TRUE, print title on new line.
newline_pre	Logical: If TRUE, print newline before list.
format_fn_rhs	Formatting function for right-hand side values.
print_class	Logical: If TRUE, print abbreviated class of object.
abbrev_class_n	Integer: Number of characters to abbreviate class names to.
print_df	Logical: If TRUE, print data frame contents, otherwise print n rows and columns.
print_S4	Logical: If TRUE, print S4 object contents, otherwise print class name.
limit	Integer: Maximum number of items to show. Use -1 for unlimited.

Details

Data frames in R began life as lists

Author(s)

EDG

qstat	SGE <i>qstat</i>
-------	------------------

Description

Run SGE qstat

Usage

```
qstat()
```

Details

alias for `system("qstat")`

Value

Called for its side effect of printing the SGE queue status.

Examples

```
## Not run:  
qstat()  
  
## End(Not run)
```

`recycle`

Recycle values of vector to match length of target

Description

Recycle values of vector to match length of target

Usage

```
recycle(x, target)
```

Arguments

<code>x</code>	Vector to be recycled
<code>target</code>	Object whose length defines target length

Details

Used internally by many functions.

Value

Vector.

Author(s)

EDG

`repr`

String representation

Description

String representation

Usage

```
repr(x, ...)
```

Details

Exported as internal function for use by other rtemis packages.

Value

Character string representation of the object.

Author(s)

EDG

<code>repr_ls</code>	<i>Show list as formatted string</i>
----------------------	--------------------------------------

Description

Works exactly like printls, but instead of printing to console with cat, it outputs a single string, formatted using mformat, so that cat(repr_ls(x)) looks identical to printls(x) for any list x

Usage

```
repr_ls(  
  x,  
  prefix = "",  
  pad = 2L,  
  item_format = bold,  
  maxlen = 4L,  
  center_title = TRUE,  
  title = NULL,  
  title_newline = TRUE,  
  newline_pre = FALSE,  
  format_fn_rhs = ddSci,  
  print_class = TRUE,  
  abbrev_class_n = 3L,  
  print_df = FALSE,  
  print_S4 = FALSE,  
  limit = 12L,  
  output_type = c("ansi", "html", "plain")  
)
```

Arguments

<code>x</code>	list or object that will be converted to a list.
<code>prefix</code>	Character: Optional prefix for names.
<code>pad</code>	Integer: Pad output with this many spaces.
<code>item_format</code>	Formatting function for items.
<code>maxlen</code>	Integer: Maximum length of items to show using headdot() before truncating with ellipsis.
<code>center_title</code>	Logical: If TRUE, autopad title for centering, if present.
<code>title</code>	Character: Title to print before list.
<code>title_newline</code>	Logical: If TRUE, print title on new line.
<code>newline_pre</code>	Logical: If TRUE, print newline before list.
<code>format_fn_rhs</code>	Formatting function for right-hand side of items.
<code>print_class</code>	Logical: If TRUE, print abbreviated class of object.
<code>abbrev_class_n</code>	Integer: Number of characters to abbreviate class names to.
<code>print_df</code>	Logical: If TRUE, print data frame contents, otherwise print n rows and columns.
<code>print_S4</code>	Logical: If TRUE, print S4 object contents, otherwise print class name.
<code>limit</code>	Integer: Maximum number of items to show.
<code>output_type</code>	Character: Output type for mformat ("ansi", "html", "plain").

Details

Exported as internal function for use by other rtemis packages.

Value

Character: Formatted string that can be printed with cat()

Author(s)

EDG

repr_S7name

Show S7 class name

Description

Show S7 class name

Usage

```
repr_S7name(x, col = col_object, pad = 0L, prefix = NULL, output_type = NULL)
```

Arguments

x	Character: S7 class name.
col	Color: Color code for the object name.
pad	Integer: Number of spaces to pad the message with.
prefix	Character: Prefix to add to the object name.
output_type	Character: Output type ("ansi", "html", "plain").

Value

Character: Formatted string that can be printed with cat().

Author(s)

EDG

Examples

```
repr_S7name("Supervised") |> cat()
```

rnormmat*Random Normal Matrix*

Description

Create a matrix or data frame of defined dimensions, whose columns are random normal vectors

Usage

```
rnormmat(  
  nrow = 10,  
  ncol = 10,  
  mean = 0,  
  sd = 1,  
  return_df = FALSE,  
  seed = NULL  
)
```

Arguments

nrow	Integer: Number of rows.
ncol	Integer: Number of columns.
mean	Float: Mean.
sd	Float: Standard deviation.
return_df	Logical: If TRUE, return data.frame, otherwise matrix.
seed	Integer: Set seed for rnorm.

Value

matrix or data.frame.

Author(s)

EDG

Examples

```
x <- rnormmat(20, 5, mean = 12, sd = 6, return_df = TRUE, seed = 2026)  
x
```

rtemis_colors *rtemis Color System*

Description

A named list of colors used consistently across all packages in the rtemis ecosystem.

Usage

```
rtemis_colors
```

Format

A named list with the following elements:

```
rt_red "kaimana red"
rt_blue "kaimana light blue"
rt_green "kaimana medium green"
rt_orange "genlib orange"
rt_teal "rtemis teal"
rt_purple "rtemis purple"
rt_magenta "rtemis magenta"
highlight_col "rtemis teal"
col_object "rtemis teal"
col_info "lmd burgundy"
col_outer "kaimana red"
col_tuner "genlib orange"
```

Details

Colors are provided as hex strings.

Author(s)

EDG

Examples

```
rtemis_colors[["rt_teal"]]
```

rtpalette*Color Palettes*

Description

`rtpalette()` prints names of available color palettes. Each palette is a named list of hexadecimal color definitions which can be used with any graphics function. `rtpalette(palette_name)` returns a list of colors for a given palette.

Usage

```
rtpalette(palette = NULL, verbosity = 1L)
```

Arguments

palette	Character: Name of palette to return. Default = NULL: available palette names are printed and no palette is returned.
verbosity	Integer: Verbosity level.

Value

A list of available palettes, invisibly.

Author(s)

EDG

Examples

```
# Print available palettes
rtpalette()
# Get the Imperial palette
rtpalette("imperial")
```

rtversion*Get rtemis version and system info*

Description

Get rtemis version and system info

Usage

```
rtversion()
```

Value

List: rtemis version and system info, invisibly.

Author(s)

EDG

Examples

```
rtversion()
```

rt_letters*Construct an n-length vector of letters***Description**

Returns an n-length vector of the latin alphabet, replicating for every 26 characters

Usage

```
rt_letters(n = 100, caps = FALSE)
```

Arguments

- | | |
|------|-----------------------------------|
| n | Length of vector to return |
| caps | Logical: If TRUE, return all caps |

rt_reactable*View table using reactable***Description**

View table using reactable

Usage

```
rt_reactable(
  x,
  datatypes = NULL,
  lightsout = TRUE,
  bg = "#121212",
  pagination = TRUE,
  searchable = TRUE,
  bordered = TRUE,
  ...
)
```

Arguments

x	data.frame, data.table or similar
datatypes	Character vector: Data types of columns in x, e.g. c("numeric", "factor", "character")
lightsout	Logical: If TRUE, use dark theme.
bg	Background color.
pagination	Logical: If TRUE, paginate table.
searchable	Logical: If TRUE, add search box.
bordered	Logical: If TRUE, add border.
...	Additional arguments passed to reactable::reactable

Value

reactable object.

Author(s)

E D Gennatas

Examples

```
## Not run:
# needs html viewer
rt_reactable(iris, datatypes = sapply(iris, class))

## End(Not run)
```

rules2medmod

Convert rules from cutoffs to median/mode and range

Description

Convert rules from cutoffs to median (range) and mode (range) format

Usage

```
rules2medmod(rules, x, .ddSci = TRUE, verbosity = 1L)
```

Arguments

rules	Character, vector: Input rules
x	Data frame: Data to evaluate rules
.ddSci	Logical: If TRUE, format all continuous variables using <code>ddSci</code> , which will give either 2 decimal places, or scientific notation if two decimal places result in 0.00
verbosity	Integer: Verbosity level.

Value

Character vector.

Author(s)

EDG

runifmat*Random Uniform Matrix***Description**

Create a matrix or data frame of defined dimensions, whose columns are random uniform vectors

Usage

```
runifmat(
  nrow = 10,
  ncol = 10,
  min = 0,
  max = 1,
  return_df = FALSE,
  seed = NULL
)
```

Arguments

<code>nrow</code>	Integer: Number of rows.
<code>ncol</code>	Integer: Number of columns.
<code>min</code>	Float: Min.
<code>max</code>	Float: Max.
<code>return_df</code>	Logical: If TRUE, return data.frame, otherwise matrix.
<code>seed</code>	Integer: Set seed for rnorm.

Value

matrix or data.frame.

Author(s)

EDG

Examples

```
x <- runifmat(20, 5, min = 12, max = 18, return_df = TRUE, seed = 2026)
x
```

setdiffsym	<i>Symmetric Set Difference</i>
------------	---------------------------------

Description

Symmetric Set Difference

Usage

```
setdiffsym(x, y)
```

Arguments

x	vector
y	vector of same type as x

Value

Vector.

Author(s)

EDG

Examples

```
setdiff(1:10, 1:5)
setdiff(1:5, 1:10)
setdiffsym(1:10, 1:5)
setdiffsym(1:5, 1:10)
```

sge_submit	<i>Submit expression to SGE grid</i>
------------	--------------------------------------

Description

Submit expression to SGE grid

Usage

```
sge_submit(
  expr,
  obj_names = NULL,
  packages = NULL,
  queue = NULL,
  n_workers = 4,
  sge_out = file.path(getwd(), "./sge_out"),
  sge_error = sge_out,
  sge_env = "#! /usr/bin/env bash",
  sge_opts = "#$ -cwd",
```

```
R_command = NULL,
system_command = NULL,
h_rt = "00:25:00",
mem_free = NULL,
temp_dir = file.path(getwd(), ".sge_tempdir"),
verbosity = 1L
)
```

Arguments

<code>expr</code>	R expression
<code>obj_names</code>	Character vector: Names of objects to copy to cluster R session
<code>packages</code>	Character vector: Names of packages to load in cluster R session
<code>queue</code>	Character: Name of SGE queue to submit to
<code>n_workers</code>	Integer: Number of threads to request from scheduler
<code>sge_out</code>	Character: Path to directory to write standard out message files
<code>sge_error</code>	Character: Path to directory to write error message files
<code>sge_env</code>	Character: Shell environment for script to be submitted to SGE
<code>sge_opts</code>	Character: SGE options that will be written in shell script. Default = "#\$ -cwd"
<code>R_command</code>	Character: Optional R command(s) to run at the beginning of the R script
<code>system_command</code>	Character: system command to be run by shell script before executing R code. For example a command that export the R executable to use
<code>h_rt</code>	Character: Max time to request. Default = "00:25:00", i.e. 25 minutes
<code>mem_free</code>	Character: Amount of memory to request from the scheduler
<code>temp_dir</code>	Character: Temporary directory that is accessible to all execution nodes. Default = <code>file.path(getwd(), ".sge_tempdir")</code> You can use <code>tempdir()</code> if all execution nodes have access to the same filesystem as the submit node.
<code>verbosity</code>	Integer: Verbosity level.

Value

Character, invisibly: The command that was submitted to SGE.

Author(s)

EDG

Examples

```
## Not run:
sge_submit({
  # Your code here
}, obj_names = c("df1", "model1"), packages = c("rtemis", "data.table"),
queue = "all.q", n_workers = 4, h_rt = "01:00:00", mem_free = "4G")

## End(Not run)
```

show_pad	<i>Add padding</i>
----------	--------------------

Description

Convenience function to add padding.

Usage

```
show_pad(pad = 2L, output_type = NULL)
```

Arguments

pad	Integer: Number of spaces to output - that's all.
output_type	Character: Output type ("ansi", "html", "plain").

Author(s)

EDG

show_range	<i>Print range of continuous variable</i>
------------	---

Description

Print range of continuous variable

Usage

```
show_range(x, ddSci = TRUE, decimal_places = 1, na.rm = TRUE)
```

Arguments

x	Numeric vector
ddSci	Logical: If TRUE, use <code>ddSci</code> or range.
decimal_places	Integer: Number of decimal place to use if ddSci = TRUE.
na.rm	Logical: passed to base::range

Value

Called for its side effect of printing the range of x.

Author(s)

EDG

size	<i>Size of object</i>
-------------	-----------------------

Description

Returns the size of an object

Usage

```
size(x, verbosity = 1L)
```

Arguments

- | | |
|------------------|---|
| x | any object with <code>length()</code> or <code>dim()</code> . |
| verbosity | Integer: Verbosity level. If > 0, print size to console |

Details

If `dim(x)` is NULL, returns `length(x)`.

Value

Integer vector with length equal to the number of dimensions of `x`, invisibly.

Author(s)

EDG

Examples

```
x <- rnorm(20)
size(x)
# 20
x <- matrix(rnorm(100), 20, 5)
size(x)
# 20 5
```

table_column_attr	<i>Tabulate column attributes</i>
--------------------------	-----------------------------------

Description

Tabulate column attributes

Usage

```
table_column_attr(x, attr = "source", useNA = "always")
```

Arguments

x	data.frame or similar: Input data set.
attr	Character: Attribute to get
useNA	Character: Passed to table

Value

table.

Author(s)

EDG

Examples

```
library(data.table)
x <- data.table(
  id = 1:5,
  sbp = rnorm(5, 120, 15),
  dbp = rnorm(5, 80, 10),
  paO2 = rnorm(5, 90, 10),
  paCO2 = rnorm(5, 40, 5)
)
setattr(x[["sbp"]], "source", "outpatient")
setattr(x[["dbp"]], "source", "outpatient")
setattr(x[["paO2"]], "source", "icu")
setattr(x[["paCO2"]], "source", "icu")
table_column_attr(x, "source")
```

thin

Make text thin/light

Description

A `fmt()` convenience wrapper for making text thin/light.

Usage

```
thin(text, output_type = c("ansi", "html", "plain"))
```

Arguments

text	Character: Text to make thin
output_type	Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with thin/light styling

Author(s)

EDG

<code>underline</code>	<i>Make text underlined</i>
------------------------	-----------------------------

Description

A `fmt()` convenience wrapper for making text underlined.

Usage

```
underline(text, output_type = c("ansi", "html", "plain"))
```

Arguments

<code>text</code>	Character: Text to underline
<code>output_type</code>	Character: Output type ("ansi", "html", "plain")

Value

Character: Formatted text with underline styling

Author(s)

EDG

<code>uniprot_get</code>	<i>Get protein sequence from UniProt</i>
--------------------------	--

Description

Get protein sequence from UniProt

Usage

```
uniprot_get(
  accession,
  baseURL = "https://rest.uniprot.org/uniprotkb",
  verbosity = 1
)
```

Arguments

<code>accession</code>	Character: UniProt Accession number - e.g. "Q9UMX9"
<code>baseURL</code>	Character: UniProt rest API base URL. Default = "https://rest.uniprot.org/uniprotkb"
<code>verbosity</code>	Integer: Verbosity level.

Value

List with three elements: Identifier, Annotation, and Sequence.

Author(s)

E.D. Gennatas

Examples

```
## Not run:  
mapt <- uniprot_get("Q9UMX9")  
  
## End(Not run)
```

uniquevalsperfeat *Unique values per feature*

Description

Get number of unique values per features

Usage

```
uniquevalsperfeat(x, excludeNA = FALSE)
```

Arguments

x	matrix or data frame input
excludeNA	Logical: If TRUE, exclude NA values from unique count.

Value

Vector, integer of length NCOL(x) with number of unique values per column/feature

Author(s)

EDG

Examples

```
## Not run:  
uniquevalsperfeat(iris)  
  
## End(Not run)
```

vec2df	<i>Vector to data.frame</i>
--------	-----------------------------

Description

Convert vector to 1-row data.frame, maintaining names if present

Usage

```
vec2df(x, col_names = NULL)
```

Arguments

x	Vector.
col_names	Character: Name of the vector.

Value

data.frame.

Author(s)

EDG

xtdescribe	<i>Describe longitudinal dataset</i>
------------	--------------------------------------

Description

This function emulates the *xtdescribe* function in Stata.

Usage

```
xtdescribe(x, id_col = 1, time_col = 2, n_patterns = 9)
```

Arguments

x	data.frame: Longitudinal data with ID and time variables.
id_col	Integer: The column position of the ID variable.
time_col	Integer: The column position of the time variable.
n_patterns	Integer: The number of patterns to display.

Value

data.frame: Summary of participation patterns, returned invisibly.

Author(s)

EDG

Examples

```
## Not run:  
# Load example longitudinal dataset  
data(xt_example, package = "rtemis")  
  
# Describe the longitudinal structure  
xtdescribe(xt_example)  
  
## End(Not run)
```

Index

- * datasets
 - rtemis_colors, 64
- * internal
 - check_inherits, 6
 - highlight, 40
 - printdf, 57
 - printls, 58
 - repr, 60
 - repr_ls, 61
 - repr_S7name, 62
- ansi256_to_hex, 4
- bold, 4
- check_dependencies, 5
- check_inherits, 6
- clean_colnames, 6
- clean_int, 7
- clean_names, 8
- col256, 8
- col2grayscale, 9
- col2hex, 10
- color_adjust, 13
- color_fade, 13
- color_invertRGB, 14
- color_mix, 14
- color_op, 15
- color_txt_columns, 16
- colorgrad, 10
- ddSci, 17, 58, 67, 71
- desaturate, 18
- df_movecolumn, 18
- drange, 19
- dt_describe, 20
- dt_inspect_types, 21
- dt_keybin_reshape, 21
- dt_merge, 22
- dt_names_by_attr, 24
- dt_nunique_perfeat, 24
- dt_pctmatch, 25
- dt_pctmissing, 26
- dt_set_autotypes, 26
- dt_set_clean_all, 28
- dt_set_cleanfactorlevels, 27
- dt_set_logical2factor, 29
- export_plotly, 30
- factor_NA2missing, 31
- fct_describe, 31
- filter_order, 32
- fmt, 33
- fmt_gradient, 34
- get_loaded_pkg_version, 36
- get_mode, 37
- get_output_type, 37
- get_vars_from_rules, 38
- getcharacternames (getnames), 35
- getdatenames (getnames), 35
- getfactornames (getnames), 35
- getlogicalnames (getnames), 35
- getnames, 35
- getnamesandtypes, 36
- getnumericnames (getnames), 35
- graph_node_metrics, 39
- gray, 39
- green, 40
- highlight, 40
- iflengthy, 41
- index_col_by_attr, 42
- init_project_dir, 42
- inspect_type, 21, 43
- is_constant, 44
- is_discrete, 44
- italic, 45
- labelify, 45
- list2csv, 46
- lotri2edgeList, 47
- make_path, 48
- matchcases, 48
- mgetnames, 50
- msg, 51

msg0 (msg), 51
msgdatetime, 52
msgdone, 52
msgstart, 53
muted, 53

names_by_class, 54

orange, 55

path.expand, 48
pcat, 55
previewcolor, 56
printfdf, 57
printls, 58

qstat, 59

recycle, 60
repr, 60
repr_ls, 61
repr_S7name, 62
rnormmat, 63
rt_letters, 66
rt_reactable, 66
rtemis_colors, 64
rtemisutils (rtemisutils-package), 3
rtemisutils-package, 3
rtpalette, 65
rtversion, 65
rules2medmod, 67
runifmat, 68

setdiffsym, 69
sge_submit, 69
show_pad, 71
show_range, 71
size, 72

table_column_attr, 72
thin, 73

underline, 74
uniprot_get, 74
uniquevalsperfeat, 75

vec2df, 76

xtdescribe, 76