Package 'radiant.data'

October 23, 2024

Title Data Menu for Radiant: Business Analytics using R and Shiny

Version 1.6.7

Date 2024-10-22

Description The Radiant Data menu includes interfaces for loading, saving, viewing, visualizing, summarizing, transforming, and combining data. It also contains functionality to generate reproducible reports of the analyses conducted in the application.

Depends R (>= 4.3.0), magrittr (>= 1.5), ggplot2 (>= 3.4.2), lubridate (>= 1.7.4), tidyr (>= 0.8.2), dplyr (>= 1.1.2)

Imports tibble (>= 1.4.2), rlang (>= 0.4.10), broom (>= 0.5.2), car (>= 3.0-0), knitr (>= 1.20), markdown (>= 1.7), rmarkdown(>= 2.22), shiny (>= 1.8.1), jsonlite (>= 1.0), shinyAce (>= 0.4.1), psych (>= 1.8.4), DT (>= 0.28), readr (>= 1.1.1), readxl (>= 1.0.0), writexl (>= 0.2), scales (>= 0.4.0), curl (>= 2.5), rstudioapi (>= 0.7), import (>= 1.1.0), plotly (>= 4.7.1), glue (>= 1.3.0), shinyFiles (>= 0.9.1), stringi (>= 1.2.4), randomizr (>= 0.20.0), patchwork (>= 1.0.0), bslib (>= 0.5.0), png, MASS, base64enc

Suggests arrow (>= 12.0.1), dbplyr (>= 2.1.1), DBI (>= 0.7), RSQLite (>= 2.0), RPostgres (>= 1.4.4), webshot (>= 0.5.0), testthat (>= 2.0.0), pkgdown (>= 1.1.0)

URL https://github.com/radiant-rstats/radiant.data/, https://radiant-rstats.github.io/radiant.data/, https://radiant-rstats.github.io/docs/

BugReports https://github.com/radiant-rstats/radiant.data/issues/

License AGPL-3 | file LICENSE

LazyData true

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author Vincent Nijs [aut, cre],
Niklas von Hertzen [aut] (html2canvas library)

Maintainer Vincent Nijs <radiant@rady.ucsd.edu>

Repository CRAN

Date/Publication 2024-10-23 04:20:02 UTC

Contents

`add_class` *Convenience function to add a class*

Description

Convenience function to add a class

Usage

```
add_class(x, cl)
```

Arguments

`x`	Object
`cl`	Vector of class labels to add

Examples

```
foo <- "some text" %>% add_class("text")
foo <- "some text" %>% add_class(c("text", "another class"))
```

`add_description` *Convenience function to add a markdown description to a data.frame*

Description

Convenience function to add a markdown description to a data.frame

Usage

```
add_description(df, md = "", path = "")
```

Arguments

`df`	A data.frame or tibble
`md`	Data description in markdown format
`path`	Path to a text file with the data description in markdown format

See Also

See also `register`

Examples

```
if (interactive()) {
 mt <- mtcars |> add_description(md = "# MTCARS\n\nThis data.frame contains information on ...")
  describe(mt)
}
```

`arrange_data` *Arrange data with user-specified expression*

Description

Arrange data with user-specified expression

Usage

```
arrange_data(dataset, expr = NULL)
```

Arguments

`dataset`	Data frame to arrange
`expr`	Expression to use arrange rows from the specified dataset

Details

Arrange data, likely in combination with slicing

Value

Arranged data frame

`as_character` *Wrapper for as.character*

Description

Wrapper for as.character

Usage

```
as_character(x)
```

Arguments

`x`	Input vector

as_distance *Distance in kilometers or miles between two locations based on lat-long Function based on* http://www.movable-type.co.uk/scripts/latlong.html. *Uses the haversine formula*

Description

Distance in kilometers or miles between two locations based on lat-long Function based on http://www.movable-type.co.uk/scripts/latlong.html. Uses the haversine formula

Usage

```
as_distance(
  lat1,
  long1,
  lat2,
  long2,
  unit = "km",
  R = c(km = 6371, miles = 3959)[[unit]]
)
```

Arguments

lat1	Latitude of location 1
long1	Longitude of location 1
lat2	Latitude of location 2
long2	Longitude of location 2
unit	Measure kilometers ("km", default) or miles ("miles")
R	Radius of the earth

Value

Distance between two points

Examples

```
as_distance(32.8245525, -117.0951632, 40.7033127, -73.979681, unit = "km")
as_distance(32.8245525, -117.0951632, 40.7033127, -73.979681, unit = "miles")
```

as_dmy — *Convert input in day-month-year format to date*

Description

Convert input in day-month-year format to date

Usage

```
as_dmy(x)
```

Arguments

x	Input variable

Value

Date variable of class Date

Examples

```
as_dmy("1-2-2014")
```

as_dmy_hm — *Convert input in day-month-year-hour-minute format to date-time*

Description

Convert input in day-month-year-hour-minute format to date-time

Usage

```
as_dmy_hm(x)
```

Arguments

x	Input variable

Value

Date-time variable of class Date

Examples

```
as_mdy_hm("1-1-2014 12:15")
```

as_dmy_hms — *Convert input in day-month-year-hour-minute-second format to date-time*

Description

Convert input in day-month-year-hour-minute-second format to date-time

Usage

```
as_dmy_hms(x)
```

Arguments

x	Input variable

Value

Date-time variable of class Date

Examples

```
as_mdy_hms("1-1-2014 12:15:01")
```

as_duration — *Wrapper for lubridate's as.duration function. Result converted to numeric*

Description

Wrapper for lubridate's as.duration function. Result converted to numeric

Usage

```
as_duration(x)
```

Arguments

x	Time difference

as_factor *Wrapper for factor with ordered = FALSE*

Description

Wrapper for factor with ordered = FALSE

Usage

```
as_factor(x, ordered = FALSE)
```

Arguments

x	Input vector
ordered	Order factor levels (TRUE, FALSE)

as_hm *Convert input in hour-minute format to time*

Description

Convert input in hour-minute format to time

Usage

```
as_hm(x)
```

Arguments

x	Input variable

Value

Time variable of class Period

Examples

```
as_hm("12:45")
## Not run:
as_hm("12:45") %>% minute()

## End(Not run)
```

as_hms *Convert input in hour-minute-second format to time*

Description

Convert input in hour-minute-second format to time

Usage

```
as_hms(x)
```

Arguments

x	Input variable

Value

Time variable of class Period

Examples

```
as_hms("12:45:00")
## Not run:
as_hms("12:45:00") %>% hour()
as_hms("12:45:00") %>% second()

## End(Not run)
```

as_integer *Convert variable to integer avoiding potential issues with factors*

Description

Convert variable to integer avoiding potential issues with factors

Usage

```
as_integer(x)
```

Arguments

x	Input variable

Value

Integer

Examples

```
as_integer(rnorm(10))
as_integer(letters)
as_integer(as.factor(5:10))
as.integer(as.factor(5:10))
as_integer(c("a", "b"))
as_integer(c("0", "1"))
as_integer(as.factor(c("0", "1")))
```

---

as_mdy *Convert input in month-day-year format to date*

Description

Convert input in month-day-year format to date

Usage

```
as_mdy(x)
```

Arguments

x	Input variable

Details

Use as.character if x is a factor

Value

Date variable of class Date

Examples

```
as_mdy("2-1-2014")
## Not run:
as_mdy("2-1-2014") %>% month(label = TRUE)
as_mdy("2-1-2014") %>% week()
as_mdy("2-1-2014") %>% wday(label = TRUE)

## End(Not run)
```

as_mdy_hm — *Convert input in month-day-year-hour-minute format to date-time*

Description

Convert input in month-day-year-hour-minute format to date-time

Usage

```
as_mdy_hm(x)
```

Arguments

x	Input variable

Value

Date-time variable of class Date

Examples

```
as_mdy_hm("1-1-2014 12:15")
```

as_mdy_hms — *Convert input in month-day-year-hour-minute-second format to date-time*

Description

Convert input in month-day-year-hour-minute-second format to date-time

Usage

```
as_mdy_hms(x)
```

Arguments

x	Input variable

Value

Date-time variable of class Date

Examples

```
as_mdy_hms("1-1-2014 12:15:01")
```

as_numeric *Convert variable to numeric avoiding potential issues with factors*

Description

Convert variable to numeric avoiding potential issues with factors

Usage

```
as_numeric(x)
```

Arguments

x	Input variable

Value

Numeric

Examples

```
as_numeric(rnorm(10))
as_numeric(letters)
as_numeric(as.factor(5:10))
as.numeric(as.factor(5:10))
as_numeric(c("a", "b"))
as_numeric(c("3", "4"))
as_numeric(as.factor(c("3", "4")))
```

as_ymd *Convert input in year-month-day format to date*

Description

Convert input in year-month-day format to date

Usage

```
as_ymd(x)
```

Arguments

x	Input variable

Value

Date variable of class Date

Examples

```
as_ymd("2013-1-1")
```

---

as_ymd_hm *Convert input in year-month-day-hour-minute format to date-time*

Description

Convert input in year-month-day-hour-minute format to date-time

Usage

```
as_ymd_hm(x)
```

Arguments

x	Input variable

Value

Date-time variable of class Date

Examples

```
as_ymd_hm("2014-1-1 12:15")
```

---

as_ymd_hms *Convert input in year-month-day-hour-minute-second format to date-time*

Description

Convert input in year-month-day-hour-minute-second format to date-time

Usage

```
as_ymd_hms(x)
```

Arguments

x	Input variable

Value

Date-time variable of class Date

Examples

```
as_ymd_hms("2014-1-1 12:15:01")
## Not run:
as_ymd_hms("2014-1-1 12:15:01") %>% as.Date()
as_ymd_hms("2014-1-1 12:15:01") %>% month()
as_ymd_hms("2014-1-1 12:15:01") %>% hour()

## End(Not run)
```

---

avengers *Avengers*

Description

Avengers

Usage

```
data(avengers)
```

Format

A data frame with 7 rows and 4 variables

Details

List of avengers. The dataset is used to illustrate data merging / joining. Description provided in attr(avengers,"description")

---

center *Center*

Description

Center

Usage

```
center(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

If x is a numeric variable return x - mean(x)

---

choose_dir *Choose a directory interactively*

Description

Choose a directory interactively

Usage

```
choose_dir(...)
```

Arguments

...	Arguments passed to utils::choose.dir on Windows

Details

Open a file dialog to select a directory. Uses JavaScript on Mac, utils::choose.dir on Windows, and dirname(file.choose()) on Linux

Value

Path to the directory selected by the user

Examples

```
## Not run:
choose_dir()

## End(Not run)
```

choose_files *Choose files interactively*

Description

Choose files interactively

Usage

```
choose_files(...)
```

Arguments

...	Strings used to indicate which file types should be available for selection (e.g., "csv" or "pdf")

Details

Open a file dialog. Uses JavaScript on Mac, utils::choose.files on Windows, and file.choose() on Linux

Value

Vector of paths to files selected by the user

Examples

```
## Not run:
choose_files("pdf", "csv")

## End(Not run)
```

ci_label *Labels for confidence intervals*

Description

Labels for confidence intervals

Usage

```
ci_label(alt = "two.sided", cl = 0.95, dec = 3)
```

Arguments

alt	Type of hypothesis ("two.sided","less","greater")
cl	Confidence level
dec	Number of decimals to show

Value

A character vector with labels for a confidence interval

Examples

```
ci_label("less", .95)
ci_label("two.sided", .95)
ci_label("greater", .9)
```

---

ci_perc *Values at confidence levels*

Description

Values at confidence levels

Usage

```
ci_perc(dat, alt = "two.sided", cl = 0.95)
```

Arguments

dat	Data
alt	Type of hypothesis ("two.sided","less","greater")
cl	Confidence level

Value

A vector with values at a confidence level

Examples

```
ci_perc(0:100, "less", .95)
ci_perc(0:100, "greater", .95)
ci_perc(0:100, "two.sided", .80)
```

combine_data *Combine datasets using dplyr's bind and join functions*

Description

Combine datasets using dplyr's bind and join functions

Usage

```
combine_data(
  x,
  y,
  by = "",
  add = "",
  type = "inner_join",
  data_filter = "",
  arr = "",
  rows = NULL,
  envir = parent.frame(),
  ...
)
```

Arguments

x	Dataset
y	Dataset to combine with x
by	Variables used to combine 'x' and 'y'
add	Variables to add from 'y'
type	The main bind and join types from the dplyr package are provided. **inner_join** returns all rows from x with matching values in y, and all columns from x and y. If there are multiple matches between x and y, all match combinations are returned. **left_join** returns all rows from x, and all columns from x and y. If there are multiple matches between x and y, all match combinations are returned. **right_join** is equivalent to a left join for datasets y and x. **full_join** combines two datasets, keeping rows and columns that appear in either. **semi_join** returns all rows from x with matching values in y, keeping just columns from x. A semi join differs from an inner join because an inner join will return one row of x for each matching row of y, whereas a semi join will never duplicate rows of x. **anti_join** returns all rows from x without matching values in y, keeping only columns from x. **bind_rows** and **bind_cols** are also included, as are **intersect**, **union**, and **setdiff**. See https://radiant-rstats.github.io/docs/data/combine.html for further details
data_filter	Expression used to filter the dataset. This should be a string (e.g., "price > 10000")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")

rows	Rows to select from the specified dataset
envir	Environment to extract data from
...	further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/data/combine.html for an example in Radiant

Value

Combined dataset

Examples

```
avengers %>% combine_data(superheroes, type = "bind_cols")
combine_data(avengers, superheroes, type = "bind_cols")
avengers %>% combine_data(superheroes, type = "bind_rows")
avengers %>% combine_data(superheroes, add = "publisher", type = "bind_rows")
```

---

copy_all *Source all package functions*

Description

Source all package functions

Usage

```
copy_all(.from)
```

Arguments

.from	The package to pull the function from

Details

Equivalent of source with local=TRUE for all package functions. Adapted from functions by smbache, author of the import package. See https://github.com/rticulate/import/issues/4/ for a discussion. This function will be deprecated when (if) it is included in https://github.com/rticulate/import/

Examples

```
copy_all(radiant.data)
```

copy_attr *Copy attributes from one object to another*

Description

Copy attributes from one object to another

Usage

```
copy_attr(to, from, attr)
```

Arguments

to	Object to copy attributes to
from	Object to copy attributes from
attr	Vector of attributes. If missing all attributes will be copied

copy_from *Source for package functions*

Description

Source for package functions

Usage

```
copy_from(.from, ...)
```

Arguments

.from	The package to pull the function from
...	Functions to pull

Details

Equivalent of source with local=TRUE for package functions. Written by smbache, author of the import package. See https://github.com/rticulate/import/issues/4/ for a discussion. This function will be deprecated when (if) it is included in https://github.com/rticulate/import/

Examples

```
copy_from(radiant.data, get_data)
```

cv *Coefficient of variation*

Description

Coefficient of variation

Usage

```
cv(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Coefficient of variation

Examples

```
cv(runif(100))
```

deregister *Deregister a data.frame or list in Radiant*

Description

Deregister a data.frame or list in Radiant

Usage

```
deregister(
  dataset,
  shiny = shiny::getDefaultReactiveDomain(),
  envir = r_data,
  info = r_info
)
```

Arguments

dataset	String containing the name of the data.frame to deregister
shiny	Check if function is called from a shiny application
envir	Environment to remove data from
info	Reactive list with information about available data in radiant

`describe` *Show dataset description*

Description

Show dataset description

Usage

```
describe(dataset, envir = parent.frame())
```

Arguments

`dataset`	Dataset with "description" attribute
`envir`	Environment to extract data from

Details

Show dataset description, if available, in html form in Rstudio viewer or the default browser. The description should be in markdown format, attached to a data.frame as an attribute with the name "description"

`diamonds` *Diamond prices*

Description

Diamond prices

Usage

```
data(diamonds)
```

Format

A data frame with 3000 rows and 10 variables

Details

A sample of 3,000 from the diamonds dataset bundled with ggplot2. Description provided in attr(diamonds,"description")

does_vary *Does a vector have non-zero variability?*

Description

Does a vector have non-zero variability?

Usage

```
does_vary(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Logical. TRUE is there is variability

Examples

```
summarise_all(diamonds, does_vary) %>% as.logical()
```

dtab *Method to create datatables*

Description

Method to create datatables

Usage

```
dtab(object, ...)
```

Arguments

object	Object of relevant class to render
...	Additional arguments

See Also

See dtab.data.frame to create an interactive table from a data.frame

See dtab.explore to create an interactive table from an explore object

See dtab.pivotr to create an interactive table from a pivotr object

`dtab.data.frame` *Create an interactive table to view, search, sort, and filter data*

Description

Create an interactive table to view, search, sort, and filter data

Usage

```
## S3 method for class 'data.frame'
dtab(
  object,
  vars = "",
  filt = "",
  arr = "",
  rows = NULL,
  nr = NULL,
  na.rm = FALSE,
  dec = 3,
  perc = "",
  filter = "top",
  pageLength = 10,
  dom = "",
  style = "bootstrap4",
  rownames = FALSE,
  caption = NULL,
  envir = parent.frame(),
  ...
)
```

Arguments

`object`	Data.frame to display
`vars`	Variables to show (default is all)
`filt`	Filter to apply to the specified dataset. For example "price > 10000" if dataset is "diamonds" (default is "")
`arr`	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
`rows`	Select rows in the specified dataset. For example "1:10" for the first 10 rows or "n()-10:n()" for the last 10 rows (default is NULL)
`nr`	Number of rows of data to include in the table. This function will be mainly used in reports so it is best to keep this number small
`na.rm`	Remove rows with missing values (default is FALSE)
`dec`	Number of decimal places to show. Default is no rounding (NULL)
`perc`	Vector of column names to be displayed as a percentage

filter	Show column filters in DT table. Options are "none", "top", "bottom"
pageLength	Number of rows to show in table
dom	Table control elements to show on the page. See https://datatables.net/reference/option/dom
style	Table formatting style ("bootstrap" or "default")
rownames	Show data.frame rownames. Default is FALSE
caption	Table caption
envir	Environment to extract data from
...	Additional arguments

Details

View, search, sort, and filter a data.frame. For styling options see https://rstudio.github.io/DT/functions.html

Examples

```
## Not run:
dtab(mtcars)

## End(Not run)
```

---

dtab.explore *Make an interactive table of summary statistics*

Description

Make an interactive table of summary statistics

Usage

```
## S3 method for class 'explore'
dtab(
  object,
  dec = 3,
  searchCols = NULL,
  order = NULL,
  pageLength = NULL,
  caption = NULL,
  ...
)
```

Arguments

object	Return value from explore
dec	Number of decimals to show
searchCols	Column search and filter
order	Column sorting
pageLength	Page length
caption	Table caption
...	further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/data/explore.html for an example in Radiant

See Also

pivotr to create a pivot table

summary.pivotr to show summaries

Examples

```
## Not run:
tab <- explore(diamonds, "price:x") %>% dtab()
tab <- explore(diamonds, "price", byvar = "cut", fun = c("n_obs", "skew"), top = "byvar") %>%
  dtab()

## End(Not run)
```

---

dtab.pivotr *Make an interactive pivot table*

Description

Make an interactive pivot table

Usage

```
## S3 method for class 'pivotr'
dtab(
  object,
  format = "none",
  perc = FALSE,
  dec = 3,
  searchCols = NULL,
  order = NULL,
```

```
  pageLength = NULL,
  caption = NULL,
  ...
)
```

Arguments

object	Return value from pivotr
format	Show Color bar ("color_bar"), Heat map ("heat"), or None ("none")
perc	Display numbers as percentages (TRUE or FALSE)
dec	Number of decimals to show
searchCols	Column search and filter
order	Column sorting
pageLength	Page length
caption	Table caption
...	further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/data/pivotr.html for an example in Radiant

See Also

pivotr to create the pivot table

summary.pivotr to print the table

Examples

```
## Not run:
pivotr(diamonds, cvars = "cut") %>% dtab()
pivotr(diamonds, cvars = c("cut", "clarity")) %>% dtab(format = "color_bar")
pivotr(diamonds, cvars = c("cut", "clarity"), normalize = "total") %>%
  dtab(format = "color_bar", perc = TRUE)

## End(Not run)
```

---

empty_level *Convert categorical variables to factors and deal with empty/missing values*

Description

Convert categorical variables to factors and deal with empty/missing values

Usage

```
empty_level(x)
```

Arguments

x	Categorical variable used in table

Value

Variable with updated levels

---

explore *Explore and summarize data*

Description

Explore and summarize data

Usage

```
explore(
  dataset,
  vars = "",
  byvar = "",
  fun = c("mean", "sd"),
  top = "fun",
  tabfilt = "",
  tabsort = "",
  tabslice = "",
  nr = Inf,
  data_filter = "",
  arr = "",
  rows = NULL,
  envir = parent.frame()
)
```

Arguments

dataset	Dataset to explore
vars	(Numeric) variables to summarize
byvar	Variable(s) to group data by
fun	Functions to use for summarizing
top	Use functions ("fun"), variables ("vars"), or group-by variables as column headers
tabfilt	Expression used to filter the table (e.g., "Total > 10000")

tabsort	Expression used to sort the table (e.g., "desc(Total)")
tabslice	Expression used to filter table (e.g., "1:5")
nr	Number of rows to display
data_filter	Expression used to filter the dataset before creating the table (e.g., "price > 10000")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

Details

See https://radiant-rstats.github.io/docs/data/explore.html for an example in Radiant

Value

A list of all variables defined in the function as an object of class explore

See Also

See summary.explore to show summaries

Examples

```
explore(diamonds, c("price", "carat")) %>% str()
explore(diamonds, "price:x")$tab
explore(diamonds, c("price", "carat"), byvar = "cut", fun = c("n_missing", "skew"))$tab
```

---

filter_data *Filter data with user-specified expression*

Description

Filter data with user-specified expression

Usage

```
filter_data(dataset, filt = "", drop = TRUE)
```

Arguments

dataset	Data frame to filter
filt	Filter expression to apply to the specified dataset
drop	Drop unused factor levels after filtering (default is TRUE)

Details

Filters can be used to view a sample from a selected dataset. For example, runif(nrow(.)) > .9 could be used to sample approximately 10

Value

Filtered data frame

Examples

```
select(diamonds, 1:3) %>% filter_data(filt = "price > max(.$price) - 100")
select(diamonds, 1:3) %>% filter_data(filt = "runif(nrow(.)) > .995")
```

`find_dropbox` *Find Dropbox folder*

Description

Find Dropbox folder

Usage

```
find_dropbox(account = 1)
```

Arguments

`account`	Integer. If multiple accounts exist, specify which one to use. By default, the first account listed is used

Details

Find the path for Dropbox if available

Value

Path to Dropbox account

`find_gdrive` *Find Google Drive folder*

Description

Find Google Drive folder

Usage

```
find_gdrive()
```

Details

Find the path for Google Drive if available

Value

Path to Google Drive folder

`find_home` *Find user directory*

Description

Find user directory

Usage

```
find_home()
```

Details

Returns /Users/x and not /Users/x/Documents

find_project *Find the Rstudio project folder*

Description

Find the Rstudio project folder

Usage

```
find_project(mess = TRUE)
```

Arguments

mess	Show or hide messages (default mess = TRUE)

Details

Find the path for the Rstudio project folder if available. The returned path is normalized (see normalizePath)

Value

Path to Rstudio project folder if available or else and empty string. The returned path is normalized

fix_names *Ensure column names are valid*

Description

Ensure column names are valid

Usage

```
fix_names(x, lower = FALSE)
```

Arguments

x	Data.frame or vector of (column) names
lower	Set letters to lower case (TRUE or FALSE)

Details

Remove symbols, trailing and leading spaces, and convert to valid R column names. Opinionated version of make.names

Examples

```
fix_names(c(" var-name ", "$amount spent", "100"))
```

fix_smart *Replace smart quotes etc.*

Description

Replace smart quotes etc.

Usage

```
fix_smart(text, all = FALSE)
```

Arguments

text	Text to be parsed
all	Should all non-ascii characters be removed? Default is FALSE

flip *Flip the DT table to put Function, Variable, or Group by on top*

Description

Flip the DT table to put Function, Variable, or Group by on top

Usage

```
flip(expl, top = "fun")
```

Arguments

expl	Return value from explore
top	The variable (type) to display at the top of the table ("fun" for Function, "var" for Variable, and "byvar" for Group by. "fun" is the default

Details

See https://radiant-rstats.github.io/docs/data/explore.html for an example in Radiant

See Also

explore to calculate summaries

summary.explore to show summaries

dtab.explore to create the DT table

Examples

```
explore(diamonds, "price:x", top = "var") %>% summary()
explore(diamonds, "price", byvar = "cut", fun = c("n_obs", "skew"), top = "byvar") %>% summary()
```

format_df *Format a data.frame with a specified number of decimal places*

Description

Format a data.frame with a specified number of decimal places

Usage

```
format_df(tbl, dec = NULL, perc = FALSE, mark = "", na.rm = FALSE, ...)
```

Arguments

tbl	Data.frame
dec	Number of decimals to show
perc	Display numbers as percentages (TRUE or FALSE)
mark	Thousand separator
na.rm	Remove missing values
...	Additional arguments for format_nr

Value

Data.frame for printing

Examples

```
data.frame(x = c("a", "b"), y = c(1L, 2L), z = c(-0.0005, 3)) %>%
  format_df(dec = 4)
data.frame(x = c(1L, 2L), y = c(0.06, 0.8)) %>%
  format_df(dec = 2, perc = TRUE)
data.frame(x = c(1L, 2L, NA), y = c(NA, 1.008, 2.8)) %>%
  format_df(dec = 2)
```

format_nr *Format a number with a specified number of decimal places, thousand sep, and a symbol*

Description

Format a number with a specified number of decimal places, thousand sep, and a symbol

Usage

```
format_nr(x, sym = "", dec = 2, perc = FALSE, mark = ",", na.rm = TRUE, ...)
```

Arguments

x	Number or vector
sym	Symbol to use
dec	Number of decimals to show
perc	Display number as a percentage
mark	Thousand separator
na.rm	Remove missing values
...	Additional arguments passed to formatC

Value

Character (vector) in the desired format

Examples

```
format_nr(2000, "$")
format_nr(2000, dec = 4)
format_nr(.05, perc = TRUE)
format_nr(c(.1, .99), perc = TRUE)
format_nr(data.frame(a = c(.1, .99)), perc = TRUE)
format_nr(data.frame(a = 1:10), sym = "$", dec = 0)
format_nr(c(1, 1.9, 1.008, 1.00))
format_nr(c(1, 1.9, 1.008, 1.00), drop0trailing = TRUE)
format_nr(NA)
format_nr(NULL)
```

---

get_class *Get variable class*

Description

Get variable class

Usage

```
get_class(dat)
```

Arguments

dat	Dataset to evaluate

Details

Get variable class information for each column in a data.frame

Value

Vector with class information for each variable

Examples

```
get_class(mtcars)
```

`get_data` *Select variables and filter data*

Description

Select variables and filter data

Usage

```
get_data(
  dataset,
  vars = "",
  filt = "",
  arr = "",
  rows = NULL,
  data_view_rows = NULL,
  na.rm = TRUE,
  rev = FALSE,
  envir = c()
)
```

Arguments

`dataset`	Dataset or name of the data.frame
`vars`	Variables to extract from the data.frame
`filt`	Filter to apply to the specified dataset
`arr`	Expression to use to arrange (sort) the specified dataset
`rows`	Select rows in the specified dataset
`data_view_rows`	Vector of rows to select. Only used by Data > View in Radiant. Users should use "rows" instead
`na.rm`	Remove rows with missing values (default is TRUE)
`rev`	Reverse filter and row selection (i.e., get the remainder)
`envir`	Environment to extract data from

Details

Function is used in radiant to select variables and filter data based on user input in string form

Value

Data.frame with specified columns and rows

Examples

```
get_data(mtcars, vars = "cyl:vs", filt = "mpg > 25")
get_data(mtcars, vars = c("mpg", "cyl"), rows = 1:10)
get_data(mtcars, vars = c("mpg", "cyl"), arr = "desc(mpg)", rows = "1:5")
```

---

get_summary *Create data.frame summary*

Description

Create data.frame summary

Usage

```
get_summary(dataset, dc = get_class(dataset), dec = 3)
```

Arguments

dataset	Data.frame
dc	Class for each variable
dec	Number of decimals to show

Details

Used in Radiant's Data > Transform tab

---

ggplotly *Work around to avoid (harmless) messages from ggplotly*

Description

Work around to avoid (harmless) messages from ggplotly

Usage

```
ggplotly(...)
```

Arguments

...	Arguments to pass to the ggplotly function in the plotly package

See Also

See the ggplotly function in the plotly package for details (?plotly::ggplotly)

indexr *Find index corrected for missing values and filters*

Description

Find index corrected for missing values and filters

Usage

```
indexr(dataset, vars = "", filt = "", arr = "", rows = NULL, cmd = "")
```

Arguments

dataset	Dataset
vars	Variables to select
filt	Data filter
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Selected rows
cmd	A command used to customize the data

install_webshot *Install webshot and phantomjs*

Description

Install webshot and phantomjs

Usage

```
install_webshot()
```

inverse *Calculate inverse of a variable*

Description

Calculate inverse of a variable

Usage

```
inverse(x)
```

Arguments

x	Input variable

Value

1/x

is.empty *Is a variable empty*

Description

Is a variable empty

Usage

```
is.empty(x, empty = "\\s*")
```

Arguments

x	Character value to evaluate
empty	Indicate what 'empty' means. Default is empty string (i.e., "")

Details

Is a variable empty

Value

TRUE if empty, else FALSE

Examples

```
is.empty("")
is.empty(NULL)
is.empty(NA)
is.empty(c())
is.empty("none", empty = "none")
is.empty("")
is.empty("   ")
is.empty(" something  ")
is.empty(c("", "something"))
is.empty(c(NA, 1:100))
is.empty(mtcars)
```

---

is_double *Is input a double (and not a date type)?*

Description

Is input a double (and not a date type)?

Usage

```
is_double(x)
```

Arguments

x	Input

Value

TRUE if double and not a type of date, else FALSE

---

is_not *Convenience function for is.null or is.na*

Description

Convenience function for is.null or is.na

Usage

```
is_not(x)
```

Arguments

x	Input

Examples

```
is_not(NA)
is_not(NULL)
is_not(c())
is_not(list())
is_not(data.frame())
```

`is_string` *Is input a string?*

Description

Is input a string?

Usage

```
is_string(x)
```

Arguments

x	Input

Value

TRUE if string, else FALSE

Examples

```
is_string("   ")
is_string("data")
is_string(c("data", ""))
is_string(NULL)
is_string(NA)
```

`iterms` *Create a vector of interaction terms for linear and logistic regression*

Description

Create a vector of interaction terms for linear and logistic regression

Usage

```
iterms(vars, nway = 2, sep = ":")
```

Arguments

vars	Labels to use
nway	2-way (2) or 3-way (3) interaction labels to create
sep	Separator to use between variable names (e.g., :)

Value

Character vector of interaction term labels

Examples

```
paste0("var", 1:3) %>% iterms(2)
paste0("var", 1:3) %>% iterms(3)
paste0("var", 1:3) %>% iterms(2, sep = ".")
```

---

launch *Launch radiant apps*

Description

Launch radiant apps

Usage

```
launch(package = "radiant.data", run = "viewer", state, ...)
```

Arguments

package	Radiant package to start. One of "radiant.data", "radiant.design", "radiant.basics", "radiant.model", "radiant.multivariate", or "radiant"
run	Run a radiant app in an external browser ("browser"), an Rstudio window ("window"), or in the Rstudio viewer ("viewer")
state	Path to statefile to load
...	additional arguments to pass to shiny::runApp (e.g, port = 8080)

Details

See https://radiant-rstats.github.io/docs/ for radiant documentation and tutorials

Examples

```
## Not run:
launch()
launch(run = "viewer")
launch(run = "window")
launch(run = "browser")

## End(Not run)
```

level_list *Generate list of levels and unique values*

Description

Generate list of levels and unique values

Usage

```
level_list(dataset, ...)
```

Arguments

dataset	A data.frame
...	Unquoted variable names to evaluate

Examples

```
data.frame(a = c(rep("a", 5), rep("b", 5)), b = c(rep(1, 5), 6:10)) %>% level_list()
level_list(mtcars, mpg, cyl)
```

ln *Natural log*

Description

Natural log

Usage

```
ln(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	Remove missing values (default is TRUE)

Value

Natural log of vector

Examples

```
ln(runif(10, 1, 2))
```

load_clip *Load data through clipboard on Windows or macOS*

Description

Load data through clipboard on Windows or macOS

Usage

```
load_clip(delim = "\t", text, suppress = TRUE)
```

Arguments

delim	Delimiter to use (tab is the default)
text	Text input to convert to table
suppress	Suppress warnings

Details

Extract data from the clipboard into a data.frame on Windows or macOS

See Also

See the save_clip

make_arrange_cmd *Generate arrange commands from user input*

Description

Generate arrange commands from user input

Usage

```
make_arrange_cmd(expr, dataset = "")
```

Arguments

expr	Expression to use arrange rows from the specified dataset
dataset	String with dataset name

Details

Form arrange command from user input

Value

Arrange command

make_train *Generate a variable used to selected a training sample*

Description

Generate a variable used to selected a training sample

Usage

```
make_train(n = 0.7, nr = NULL, blocks = NULL, seed = 1234)
```

Arguments

n	Number (or fraction) of observations to label as training
nr	Number of rows in the dataset
blocks	A vector to use for blocking or a data.frame from which to construct a blocking vector
seed	Random seed

Value

0/1 variables for filtering

Examples

```
make_train(.5, 10)
make_train(.5, 10) %>% table()
make_train(100, 1000) %>% table()
make_train(.15, blocks = mtcars$vs) %>% table() / nrow(mtcars)
make_train(.10, blocks = iris$Species) %>% table() / nrow(iris)
make_train(.5, blocks = iris[, c("Petal.Width", "Species")]) %>% table()
```

make_vec *Convert a string of numbers into a vector*

Description

Convert a string of numbers into a vector

Usage

```
make_vec(x)
```

Arguments

x	A string of numbers that may include fractions

Examples

```
make_vec("1 2 4")
make_vec("1/2 2/3 4/5")
make_vec(0.1)
```

---

me *Margin of error*

Description

Margin of error

Usage

```
me(x, conf_lev = 0.95, na.rm = TRUE)
```

Arguments

x	Input variable
conf_lev	Confidence level. The default is 0.95
na.rm	If TRUE missing values are removed before calculation

Value

Margin of error

Examples

```
me(rnorm(100))
```

meprop *Margin of error for proportion*

Description

Margin of error for proportion

Usage

```
meprop(x, conf_lev = 0.95, na.rm = TRUE)
```

Arguments

x	Input variable
conf_lev	Confidence level. The default is 0.95
na.rm	If TRUE missing values are removed before calculation

Value

Margin of error

Examples

```
meprop(c(rep(1L, 10), rep(0L, 10)))
```

modal *Calculate the mode (modal value) and return a label*

Description

Calculate the mode (modal value) and return a label

Usage

```
modal(x, na.rm = TRUE)
```

Arguments

x	A vector
na.rm	If TRUE missing values are removed before calculation

Details

From https://www.tutorialspoint.com/r/r_mean_median_mode.htm

Examples

```
modal(c("a", "b", "b"))
modal(c(1:10, 5))
modal(as.factor(c(letters, "b")))
modal(runif(100) > 0.5)
```

`month` *Add ordered argument to lubridate::month*

Description

Add ordered argument to lubridate::month

Usage

```
month(x, label = FALSE, abbr = TRUE, ordered = FALSE)
```

Arguments

`x`	Input date vector
`label`	Month as label (TRUE, FALSE)
`abbr`	Abbreviate label (TRUE, FALSE)
`ordered`	Order factor (TRUE, FALSE)

See Also

See the `month` function in the lubridate package for additional details

---

`mutate_ext` *Add transformed variables to a data frame with the option to include a custom variable name extension*

Description

Add transformed variables to a data frame with the option to include a custom variable name extension

Usage

```
mutate_ext(.tbl, .funs, ..., .ext = "", .vars = c())
```

Arguments

.tbl	Data frame to add transformed variables to
.funs	Function(s) to apply (e.g., log)
...	Variables to transform
.ext	Extension to add for each variable
.vars	A list of columns generated by dplyr::vars(), or a character vector of column names, or a numeric vector of column positions.

Details

Wrapper for dplyr::mutate_at that allows custom variable name extensions

Examples

```
mutate_ext(mtcars, .funs = log, mpg, cyl, .ext = "_ln")
mutate_ext(mtcars, .funs = log, .ext = "_ln")
mutate_ext(mtcars, .funs = log)
mutate_ext(mtcars, .funs = log, .ext = "_ln", .vars = vars(mpg, cyl))
```

---

normalize *Normalize a variable x by a variable y*

Description

Normalize a variable x by a variable y

Usage

```
normalize(x, y)
```

Arguments

x	Input variable
y	Normalizing variable

Value

x/y

n_missing *Number of missing values*

Description

Number of missing values

Usage

```
n_missing(x, ...)
```

Arguments

x	Input variable
...	Additional arguments

Value

number of missing values

Examples

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

n_obs *Number of observations*

Description

Number of observations

Usage

```
n_obs(x, ...)
```

Arguments

x	Input variable
...	Additional arguments

Value

number of observations

Examples

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

---

p01 *Calculate percentiles*

Description

Calculate percentiles

Usage

```
p01(x, na.rm = TRUE)

p025(x, na.rm = TRUE)

p05(x, na.rm = TRUE)

p10(x, na.rm = TRUE)

p25(x, na.rm = TRUE)

p75(x, na.rm = TRUE)

p90(x, na.rm = TRUE)

p95(x, na.rm = TRUE)

p975(x, na.rm = TRUE)

p99(x, na.rm = TRUE)
```

Arguments

x	Numeric vector
na.rm	If TRUE missing values are removed before calculation

Examples

```
p01(0:100)
```

parse_path *Parse file path into useful components*

Description

Parse file path into useful components

Usage

```
parse_path(path, chr = "", pdir = getwd(), mess = TRUE)
```

Arguments

path	Path to be parsed
chr	Character to wrap around path for display
pdir	Project directory if available
mess	Print messages if Dropbox or Google Drive not found

Details

Parse file path into useful components (i.e., file name, file extension, relative path, etc.)

Examples

```
list.files(".", full.names = TRUE)[1] %>% parse_path()
```

pfun *Summarize a set of numeric vectors per row*

Description

Summarize a set of numeric vectors per row

Usage

```
pfun(..., fun, na.rm = TRUE)

psum(..., na.rm = TRUE)

pmean(..., na.rm = TRUE)

pmedian(..., na.rm = TRUE)

psd(..., na.rm = TRUE)
```

```
pvar(..., na.rm = TRUE)

pcv(..., na.rm = TRUE)

pp01(..., na.rm = TRUE)

pp025(..., na.rm = TRUE)

pp05(..., na.rm = TRUE)

pp10(..., na.rm = TRUE)

pp25(..., na.rm = TRUE)

pp75(..., na.rm = TRUE)

pp95(..., na.rm = TRUE)

pp975(..., na.rm = TRUE)

pp99(..., na.rm = TRUE)
```

Arguments

`...`	Numeric vectors of the same length
`fun`	Function to apply
`na.rm`	a logical indicating whether missing values should be removed.

Details

Calculate summary statistics of the input vectors per row (or 'parallel')

Value

A vector of 'parallel' summaries of the argument vectors.

See Also

See also `pmin` and `pmax`

Examples

```
pfun(1:10, fun = mean)
psum(1:10, 10:1)
```

pivotr *Create a pivot table*

Description

Create a pivot table

Usage

```
pivotr(
  dataset,
  cvars = "",
  nvar = "None",
  fun = "mean",
  normalize = "None",
  tabfilt = "",
  tabsort = "",
  tabslice = "",
  nr = Inf,
  data_filter = "",
  arr = "",
  rows = NULL,
  envir = parent.frame()
)
```

Arguments

dataset	Dataset to tabulate
cvars	Categorical variables
nvar	Numerical variable
fun	Function to apply to numerical variable
normalize	Normalize the table by row total, column totals, or overall total
tabfilt	Expression used to filter the table (e.g., "Total > 10000")
tabsort	Expression used to sort the table (e.g., "desc(Total)")
tabslice	Expression used to filter table (e.g., "1:5")
nr	Number of rows to display
data_filter	Expression used to filter the dataset before creating the table (e.g., "price > 10000")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

Details

Create a pivot-table. See https://radiant-rstats.github.io/docs/data/pivotr.html for an example in Radiant

Examples

```
pivotr(diamonds, cvars = "cut") %>% str()
pivotr(diamonds, cvars = "cut")$tab
pivotr(diamonds, cvars = c("cut", "clarity", "color"))$tab
pivotr(diamonds, cvars = "cut:clarity", nvar = "price")$tab
pivotr(diamonds, cvars = "cut", nvar = "price")$tab
pivotr(diamonds, cvars = "cut", normalize = "total")$tab
```

---

plot.pivotr *Plot method for the pivotr function*

Description

Plot method for the pivotr function

Usage

```
## S3 method for class 'pivotr'
plot(
  x,
  type = "dodge",
  perc = FALSE,
  flip = FALSE,
  fillcol = "blue",
  opacity = 0.5,
  ...
)
```

Arguments

x	Return value from pivotr
type	Plot type to use ("fill" or "dodge" (default))
perc	Use percentage on the y-axis
flip	Flip the axes in a plot (FALSE or TRUE)
fillcol	Fill color for bar-plot when only one categorical variable has been selected (default is "blue")
opacity	Opacity for plot elements (0 to 1)
...	further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/data/pivotr for an example in Radiant

See Also

pivotr to generate summaries

summary.pivotr to show summaries

Examples

```
pivotr(diamonds, cvars = "cut") %>% plot()
pivotr(diamonds, cvars = c("cut", "clarity")) %>% plot()
pivotr(diamonds, cvars = c("cut", "clarity", "color")) %>% plot()
```

---

prop *Calculate proportion*

Description

Calculate proportion

Usage

```
prop(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Proportion of first level for a factor and of the maximum value for numeric

Examples

```
prop(c(rep(1L, 10), rep(0L, 10)))
prop(c(rep(4, 10), rep(2, 10)))
prop(rep(0, 10))
prop(factor(c(rep("a", 20), rep("b", 10))))
```

publishers *Comic publishers*

Description

Comic publishers

Usage

```
data(publishers)
```

Format

A data frame with 3 rows and 2 variables

Details

List of comic publishers from https://stat545.com/join-cheatsheet.html. The dataset is used to illustrate data merging / joining. Description provided in attr(publishers,"description")

qscatter *Create a qscatter plot similar to Stata*

Description

Create a qscatter plot similar to Stata

Usage

```
qscatter(dataset, xvar, yvar, lev = "", fun = "mean", bins = 20)
```

Arguments

dataset	Data to plot (data.frame or tibble)
xvar	Character indicating the variable to display along the X-axis of the plot
yvar	Character indicating the variable to display along the Y-axis of the plot
lev	Level in yvar to use if yvar is of type character of factor. If lev is empty then the first level is used
fun	Summary measure to apply to both the x and y variable
bins	Number of bins to use

Examples

```
qscatter(diamonds, "price", "carat")
qscatter(titanic, "age", "survived")
```

qterms *Create a vector of quadratic and cubed terms for use in linear and logistic regression*

Description

Create a vector of quadratic and cubed terms for use in linear and logistic regression

Usage

```
qterms(vars, nway = 2)
```

Arguments

vars	Variables labels to use
nway	quadratic (2) or cubic (3) term labels to create

Value

Character vector of (regression) term labels

Examples

```
qterms(c("a", "b"), 3)
qterms(c("a", "b"), 2)
```

radiant.data *radiant.data*

Description

Launch the radiant.data app in the default web browser

Usage

```
radiant.data(state, ...)
```

Arguments

state	Path to statefile to load
...	additional arguments to pass to shiny::runApp (e.g, port = 8080)

Examples

```
## Not run:
radiant.data()
radiant.data("https://github.com/radiant-rstats/docs/raw/gh-pages/examples/demo-dvd-rnd.state.rda")
radiant.data("viewer")

## End(Not run)
```

`radiant.data-deprecated` *Deprecated function(s) in the radiant.data package*

Description

These functions are provided for compatibility with previous versions of radiant but will be removed

Usage

```
mean_rm(...)
```

Arguments

`...`	Parameters to be passed to the updated functions

Details

- Replace `mean_rm` by `mean`
- Replace `median_rm` by `median`
- Replace `min_rm` by `min`
- Replace `max_rm` by `max`
- Replace `sd_rm` by `sd`
- Replace `var_rm` by `var`
- Replace `sum_rm` by `sum`
- Replace `getdata` by `get_data`
- Replace `filterdata` by `filter_data`
- Replace `combinedata` by `combine_data`
- Replace `viewdata` by `view_data`
- Replace `toFct` by `to_fct`
- Replace `fixMS` by `fix_smart`
- Replace `rounddf` by `round_df`
- Replace `formatdf` by `format_df`
- Replace `formatnr` by `format_nr`
- Replace `getclass` by `get_class`
- Replace `is_numeric` by `is_double`
- Replace `is_empty` by `is.empty`

radiant.data_url *Start radiant.data app but do not open a browser*

Description

Start radiant.data app but do not open a browser

Usage

```
radiant.data_url(state, ...)
```

Arguments

state	Path to statefile to load
...	additional arguments to pass to shiny::runApp (e.g, port = 8080)

Examples

```
## Not run:
radiant.data_url()

## End(Not run)
```

radiant.data_viewer *Launch the radiant.data app in the Rstudio viewer*

Description

Launch the radiant.data app in the Rstudio viewer

Usage

```
radiant.data_viewer(state, ...)
```

Arguments

state	Path to statefile to load
...	additional arguments to pass to shiny::runApp (e.g, port = 8080)

Examples

```
## Not run:
radiant.data_viewer()

## End(Not run)
```

radiant.data_window *Launch the radiant.data app in an Rstudio window*

Description

Launch the radiant.data app in an Rstudio window

Usage

```
radiant.data_window(state, ...)
```

Arguments

state	Path to statefile to load
...	additional arguments to pass to shiny::runApp (e.g, port = 8080)

Examples

```
## Not run:
radiant.data_window()

## End(Not run)
```

read_files *Generate code to read a file*

Description

Generate code to read a file

Usage

```
read_files(
  path,
  pdir = "",
  type = "rmd",
  to = "",
  clipboard = TRUE,
  radiant = FALSE
)
```

Arguments

path	Path to file. If empty, a file browser will be opened
pdir	Project dir
type	Generate code for _Report > Rmd_ ("rmd") or _Report > R_ ("r")
to	Name to use for object. If empty, will use file name to derive an object name
clipboard	Return code to clipboard (not available on Linux)
radiant	Should returned code be formatted for use with other code generated by Radiant?

Details

Return code to read a file at the specified path. Will open a file browser if no path is provided

Examples

```
if (interactive()) {
  read_files(clipboard = FALSE)
}
```

---

refactor *Remove/reorder levels*

Description

Remove/reorder levels

Usage

```
refactor(x, levs = levels(x), repl = NA)
```

Arguments

x	Character or Factor
levs	Set of levels to use
repl	String (or NA) used to replace missing levels

Details

Keep only a specific set of levels in a factor. By removing levels the base for comparison in, e.g., regression analysis, becomes the first level. To relabel the base use, for example, repl = 'other'

Examples

```
refactor(diamonds$cut, c("Premium", "Ideal")) %>% head()
refactor(diamonds$cut, c("Premium", "Ideal"), "Other") %>% head()
```

register *Register a data.frame or list in Radiant*

Description

Register a data.frame or list in Radiant

Usage

```
register(
  new,
  org = "",
  descr = "",
  shiny = shiny::getDefaultReactiveDomain(),
  envir = r_data
)
```

Arguments

new	String containing the name of the data.frame to register
org	Name of the original data.frame if a (working) copy is being made
descr	Data description in markdown format
shiny	Check if function is called from a shiny application
envir	Environment to assign data to

See Also

See also add_description to add a description in markdown format to a data.frame

render *Base method used to render htmlwidgets*

Description

Base method used to render htmlwidgets

Usage

```
render(object, ...)
```

Arguments

object	Object of relevant class to render
...	Additional arguments

render.datatables *Method to render DT tables*

Description

Method to render DT tables

Usage

```
## S3 method for class 'datatables'
render(object, shiny = shiny::getDefaultReactiveDomain(), ...)
```

Arguments

object	DT table
shiny	Check if function is called from a shiny application
...	Additional arguments

render.plotly *Method to render plotly plots*

Description

Method to render plotly plots

Usage

```
## S3 method for class 'plotly'
render(object, shiny = shiny::getDefaultReactiveDomain(), ...)
```

Arguments

object	plotly object
shiny	Check if function is called from a shiny application
...	Additional arguments

round_df *Round doubles in a data.frame to a specified number of decimal places*

Description

Round doubles in a data.frame to a specified number of decimal places

Usage

```
round_df(tbl, dec = 3)
```

Arguments

tbl	Data frame
dec	Number of decimals to show

Value

Data frame with rounded doubles

Examples

```
data.frame(x = as.factor(c("a", "b")), y = c(1L, 2L), z = c(-0.0005, 3.1)) %>%
  round_df(dec = 2)
```

save_clip *Save data to clipboard on Windows or macOS*

Description

Save data to clipboard on Windows or macOS

Usage

```
save_clip(dataset)
```

Arguments

dataset	Dataset to save to clipboard

Details

Save a data.frame or tibble to the clipboard on Windows or macOS

See Also

See the load_clip

sdpop *Standard deviation for the population*

Description

Standard deviation for the population

Usage

```
sdpop(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Standard deviation for the population

Examples

```
sdpop(rnorm(100))
```

sdprop *Standard deviation for proportion*

Description

Standard deviation for proportion

Usage

```
sdprop(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Standard deviation for proportion

Examples

```
sdprop(c(rep(1L, 10), rep(0L, 10)))
```

---

se *Standard error*

Description

Standard error

Usage

```
se(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Standard error

Examples

```
se(rnorm(100))
```

---

search_data *Search for a pattern in all columns of a data.frame*

Description

Search for a pattern in all columns of a data.frame

Usage

```
search_data(dataset, pattern, ignore.case = TRUE, fixed = FALSE)
```

Arguments

dataset	Data.frame to search
pattern	String to match
ignore.case	Should search be case sensitive or not (default is FALSE)
fixed	Allow regular expressions or not (default is FALSE)

See Also

See grepl for a detailed description of the function arguments

Examples

```
publishers %>% filter(search_data(., "^m"))
```

---

seprop *Standard error for proportion*

Description

Standard error for proportion

Usage

```
seprop(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Standard error for proportion

Examples

```
seprop(c(rep(1L, 10), rep(0L, 10)))
```

---

set_attr *Alias used to add an attribute*

Description

Alias used to add an attribute

Usage

```
set_attr(x, which, value)
```

Arguments

x	Object
which	Attribute name
value	Value to set

Examples

```
foo <- data.frame(price = 1:5) %>% set_attr("description", "price set in experiment ...")
```

---

show_duplicated *Show all rows with duplicated values (not just the first or last)*

Description

Show all rows with duplicated values (not just the first or last)

Usage

```
show_duplicated(.tbl, ...)
```

Arguments

.tbl	Data frame to add transformed variables to
...	Variables used to evaluate row uniqueness

Details

If an entire row is duplicated use "duplicated" to show only one of the duplicated rows. When using a subset of variables to establish uniqueness it may be of interest to show all rows that have (some) duplicate elements

Examples

```
bind_rows(mtcars, mtcars[c(1, 5, 7), ]) %>%
  show_duplicated(mpg, cyl)
bind_rows(mtcars, mtcars[c(1, 5, 7), ]) %>%
  show_duplicated()
```

sig_stars *Add stars based on p.values*

Description

Add stars based on p.values

Usage

```
sig_stars(pval)
```

Arguments

pval	Vector of p-values

Value

A vector of stars

Examples

```
sig_stars(c(.0009, .049, .009, .4, .09))
```

slice_data *Slice data with user-specified expression*

Description

Slice data with user-specified expression

Usage

```
slice_data(dataset, expr = NULL, drop = TRUE)
```

Arguments

dataset	Data frame to slice
expr	Expression to use select rows from the specified dataset
drop	Drop unused factor levels after filtering (default is TRUE)

Details

Select only a slice of the data to work with

Value

Sliced data frame

square *Calculate square of a variable*

Description

Calculate square of a variable

Usage

```
square(x)
```

Arguments

x	Input variable

Value

x^2

sshh *Hide warnings and messages and return invisible*

Description

Hide warnings and messages and return invisible

Usage

```
sshh(...)
```

Arguments

...	Inputs to keep quite

Details

Hide warnings and messages and return invisible

Examples

```
sshh(library(dplyr))
```

sshhr *Hide warnings and messages and return result*

Description

Hide warnings and messages and return result

Usage

```
sshhr(...)
```

Arguments

...	Inputs to keep quite

Details

Hide warnings and messages and return result

Examples

```
sshhr(library(dplyr))
```

standardize *Standardize*

Description

Standardize

Usage

```
standardize(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

If x is a numeric variable return (x - mean(x)) / sd(x)

store *Method to store variables in a dataset in Radiant*

Description

Method to store variables in a dataset in Radiant

Usage

```
store(dataset, object = "deprecated", ...)
```

Arguments

dataset	Dataset
object	Object of relevant class that has information to be stored
...	Additional arguments

store.explore *Deprecated: Store method for the explore function*

Description

Deprecated: Store method for the explore function

Usage

```
## S3 method for class 'explore'
store(dataset, object, name, ...)
```

Arguments

dataset	Dataset
object	Return value from explore
name	Name to assign to the dataset
...	further arguments passed to or from other methods

Details

Return the summarized data. See https://radiant-rstats.github.io/docs/data/explore.html for an example in Radiant

See Also

explore to generate summaries

store.pivotr *Deprecated: Store method for the pivotr function*

Description

Deprecated: Store method for the pivotr function

Usage

```
## S3 method for class 'pivotr'
store(dataset, object, name, ...)
```

Arguments

dataset	Dataset
object	Return value from pivotr
name	Name to assign to the dataset
...	further arguments passed to or from other methods

Details

Return the summarized data. See https://radiant-rstats.github.io/docs/data/pivotr.html for an example in Radiant

See Also

pivotr to generate summaries

subplot *Work around to avoid (harmless) messages from subplot*

Description

Work around to avoid (harmless) messages from subplot

Usage

```
subplot(..., margin = 0.04)
```

Arguments

...	Arguments to pass to the subplot function in the plotly packages
margin	Default margin to use between plots

See Also

See the subplot in the plotly package for details (?plotly::subplot)

summary.explore *Summary method for the explore function*

Description

Summary method for the explore function

Usage

```
## S3 method for class 'explore'
summary(object, dec = 3, ...)
```

Arguments

object	Return value from explore
dec	Number of decimals to show
...	further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/data/explore.html for an example in Radiant

See Also

explore to generate summaries

Examples

```
result <- explore(diamonds, "price:x")
summary(result)
result <- explore(diamonds, "price", byvar = "cut", fun = c("n_obs", "skew"))
summary(result)
explore(diamonds, "price:x", byvar = "color") %>% summary()
```

summary.pivotr *Summary method for pivotr*

Description

Summary method for pivotr

Usage

```
## S3 method for class 'pivotr'
summary(object, perc = FALSE, dec = 3, chi2 = FALSE, shiny = FALSE, ...)
```

Arguments

object	Return value from pivotr
perc	Display numbers as percentages (TRUE or FALSE)
dec	Number of decimals to show
chi2	If TRUE calculate the chi-square statistic for the (pivot) table
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/data/pivotr.html for an example in Radiant

See Also

pivotr to create the pivot-table using dplyr

Examples

```
pivotr(diamonds, cvars = "cut") %>% summary(chi2 = TRUE)
pivotr(diamonds, cvars = "cut", tabsort = "desc(n_obs)") %>% summary()
pivotr(diamonds, cvars = "cut", tabfilt = "n_obs > 700") %>% summary()
pivotr(diamonds, cvars = "cut:clarity", nvar = "price") %>% summary()
```

---

superheroes *Super heroes*

Description

Super heroes

Usage

```
data(superheroes)
```

Format

A data frame with 7 rows and 4 variables

Details

List of super heroes from https://stat545.com/join-cheatsheet.html. The dataset is used to illustrate data merging / joining. Description provided in attr(superheroes,"description")

table2data *Create data.frame from a table*

Description

Create data.frame from a table

Usage

```
table2data(dataset, freq = tail(colnames(dataset), 1))
```

Arguments

dataset	Data.frame
freq	Column name with frequency information

Examples

```
data.frame(price = c("$200", "$300"), sale = c(10, 2)) %>% table2data()
```

titanic *Survival data for the Titanic*

Description

Survival data for the Titanic

Usage

```
data(titanic)
```

Format

A data frame with 1043 rows and 10 variables

Details

Survival data for the Titanic. Description provided in attr(titanic,"description")

to_fct *Convert characters to factors*

Description

Convert characters to factors

Usage

```
to_fct(dataset, safx = 30, nuniq = 100, n = 100)
```

Arguments

dataset	Data frame
safx	Ratio of number of rows to number of unique values
nuniq	Cutoff for number of unique values
n	Cutoff for small dataset

Details

Convert columns of type character to factors based on a set of rules. By default columns will be converted for small datasets (<= 100 rows) with more rows than unique values. For larger datasets, columns are converted only when the number of unique values is <= 100 and there are 30 or more rows in the data for every unique value

Examples

```
tibble(a = c("a", "b"), b = c("a", "a"), c = 1:2) %>% to_fct()
```

varpop *Variance for the population*

Description

Variance for the population

Usage

```
varpop(x, na.rm = TRUE)
```

Arguments

x	Input variable
na.rm	If TRUE missing values are removed before calculation

Value

Variance for the population

Examples

```
varpop(rnorm(100))
```

`varprop` *Variance for proportion*

Description

Variance for proportion

Usage

```
varprop(x, na.rm = TRUE)
```

Arguments

`x`	Input variable
`na.rm`	If TRUE missing values are removed before calculation

Value

Variance for proportion

Examples

```
varprop(c(rep(1L, 10), rep(0L, 10)))
```

`view_data` *View data in a shiny-app*

Description

View data in a shiny-app

Usage

```
view_data(
  dataset,
  vars = "",
  filt = "",
  arr = "",
  rows = NULL,
  na.rm = FALSE,
  dec = 3,
  envir = parent.frame()
)
```

Arguments

dataset	Data.frame or name of the dataframe to view
vars	Variables to show (default is all)
filt	Filter to apply to the specified dataset
arr	Expression to arrange (sort) data
rows	Select rows in the specified dataset
na.rm	Remove rows with missing values (default is FALSE)
dec	Number of decimals to show
envir	Environment to extract data from

Details

View, search, sort, etc. your data

See Also

See get_data and filter_data

Examples

```
## Not run:
view_data(mtcars)

## End(Not run)
```

visualize *Visualize data using ggplot2* https://ggplot2.tidyverse.org/

Description

Visualize data using ggplot2 https://ggplot2.tidyverse.org/

Usage

```
visualize(
  dataset,
  xvar,
  yvar = "",
  comby = FALSE,
  combx = FALSE,
  type = ifelse(is.empty(yvar), "dist", "scatter"),
  nrobs = -1,
  facet_row = ".",
  facet_col = ".",
  color = "none",
  fill = "none",
  size = "none",
  fillcol = "blue",
  linecol = "black",
  pointcol = "black",
  bins = 10,
  smooth = 1,
  fun = "mean",
  check = "",
  axes = "",
  alpha = 0.5,
  theme = "theme_gray",
  base_size = 11,
  base_family = "",
  labs = list(),
  xlim = NULL,
  ylim = NULL,
  data_filter = "",
  arr = "",
  rows = NULL,
  shiny = FALSE,
  custom = FALSE,
  envir = parent.frame()
)
```

Arguments

dataset	Data to plot (data.frame or tibble)

xvar	One or more variables to display along the X-axis of the plot
yvar	Variable to display along the Y-axis of the plot (default = "none")
comby	Combine yvars in plot (TRUE or FALSE, FALSE is the default)
combx	Combine xvars in plot (TRUE or FALSE, FALSE is the default)
type	Type of plot to create. One of Distribution ('dist'), Density ('density'), Scatter ('scatter'), Surface ('surface'), Line ('line'), Bar ('bar'), or Box-plot ('box')
nrobs	Number of data points to show in scatter plots (-1 for all)
facet_row	Create vertically arranged subplots for each level of the selected factor variable
facet_col	Create horizontally arranged subplots for each level of the selected factor variable
color	Adds color to a scatter plot to generate a 'heat map'. For a line plot one line is created for each group and each is assigned a different color
fill	Display bar, distribution, and density plots by group, each with a different color. Also applied to surface plots to generate a 'heat map'
size	Numeric variable used to scale the size of scatter-plot points
fillcol	Color used for bars, boxes, etc. when no color or fill variable is specified
linecol	Color for lines when no color variable is specified
pointcol	Color for points when no color variable is specified
bins	Number of bins used for a histogram (1 - 50)
smooth	Adjust the flexibility of the loess line for scatter plots
fun	Set the summary measure for line and bar plots when the X-variable is a factor (default is "mean"). Also used to plot an error bar in a scatter plot when the X-variable is a factor. Options are "mean" and/or "median"
check	Add a regression line ("line"), a loess line ("loess"), or jitter ("jitter") to a scatter plot
axes	Flip the axes in a plot ("flip") or apply a log transformation (base e) to the y-axis ("log_y") or the x-axis ("log_x")
alpha	Opacity for plot elements (0 to 1)
theme	ggplot theme to use (e.g., "theme_gray" or "theme_classic")
base_size	Base font size to use (default = 11)
base_family	Base font family to use (e.g., "Times" or "Helvetica")
labs	Labels to use for plots
xlim	Set limit for x-axis (e.g., c(0, 1))
ylim	Set limit for y-axis (e.g., c(0, 1))
data_filter	Expression used to filter the dataset. This should be a string (e.g., "price > 10000")
arr	Expression used to sort the data. Likely used in combination for 'rows'
rows	Rows to select from the specified dataset
shiny	Logical (TRUE, FALSE) to indicate if the function call originate inside a shiny app

custom	Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and https://ggplot2.tidyverse.org for options.
envir	Environment to extract data from

Details

See https://radiant-rstats.github.io/docs/data/visualize.html for an example in Radiant

Value

Generated plots

Examples

```
visualize(diamonds, "price:cut", type = "dist", fillcol = "red")
visualize(diamonds, "carat:cut",
  yvar = "price", type = "scatter",
  pointcol = "blue", fun = c("mean", "median"), linecol = c("red", "green")
)
visualize(diamonds,
  yvar = "price", xvar = c("cut", "clarity"),
  type = "bar", fun = "median"
)
visualize(diamonds,
  yvar = "price", xvar = c("cut", "clarity"),
  type = "line", fun = "max"
)
visualize(diamonds,
  yvar = "price", xvar = "carat", type = "scatter",
  size = "table", custom = TRUE
) + scale_size(range = c(1, 10), guide = "none")
visualize(diamonds, yvar = "price", xvar = "carat", type = "scatter", custom = TRUE) +
  labs(title = "A scatterplot", x = "price in $")
visualize(diamonds, xvar = "price:carat", custom = TRUE) %>%
  wrap_plots(ncol = 2) + plot_annotation(title = "Histograms")
visualize(diamonds,
  xvar = "cut", yvar = "price", type = "bar",
  facet_row = "cut", fill = "cut"
)
```

---

wday *Add ordered argument to lubridate::wday*

Description

Add ordered argument to lubridate::wday

Usage

```
wday(x, label = FALSE, abbr = TRUE, ordered = FALSE)
```

Arguments

x	Input date vector
label	Weekday as label (TRUE, FALSE)
abbr	Abbreviate label (TRUE, FALSE)
ordered	Order factor (TRUE, FALSE)

See Also

See the `lubridate::wday()` function in the lubridate package for additional details

---

weighted.sd *Weighted standard deviation*

Description

Weighted standard deviation

Usage

```
weighted.sd(x, wt, na.rm = TRUE)
```

Arguments

x	Numeric vector
wt	Numeric vector of weights
na.rm	Remove missing values (default is TRUE)

Details

Calculate weighted standard deviation

which.pmax *Index of the maximum per row*

Description

Index of the maximum per row

Usage

```
which.pmax(...)
```

Arguments

...	Numeric or character vectors of the same length

Details

Determine the index of the maximum of the input vectors per row. Extension of `which.max`

Value

Vector of rankings

See Also

See also `which.max` and `which.pmin`

Examples

```
which.pmax(1:10, 10:1)
which.pmax(2, 10:1)
which.pmax(mtcars)
```

which.pmin *Index of the minimum per row*

Description

Index of the minimum per row

Usage

```
which.pmin(...)
```

Arguments

...	Numeric or character vectors of the same length

Details

Determine the index of the minimum of the input vectors per row. Extension of `which.min`

Value

Vector of rankings

See Also

See also `which.min` and `which.pmax`

Examples

```
which.pmin(1:10, 10:1)
which.pmin(2, 10:1)
which.pmin(mtcars)
```

---

write_parquet *Workaround to store description file together with a parquet data file*

Description

Workaround to store description file together with a parquet data file

Usage

```
write_parquet(x, file, description = attr(x, "description"))
```

Arguments

`x`	A data frame to write to disk
`file`	Path to store parquet file
`description`	Data description

`xtile` *Split a numeric variable into a number of bins and return a vector of bin numbers*

Description

Split a numeric variable into a number of bins and return a vector of bin numbers

Usage

```
xtile(x, n = 5, rev = FALSE, type = 7)
```

Arguments

`x`	Numeric variable
`n`	number of bins to create
`rev`	Reverse the order of the bin numbers
`type`	An integer between 1 and 9 to select one of the quantile algorithms described in the help for the stats::quantile function

See Also

See quantile for a description of the different algorithm types

Examples

```
xtile(1:10, 5)
xtile(1:10, 5, rev = TRUE)
xtile(c(rep(1, 6), 7:10), 5)
```

Index