Package 'radiant.data'

November 6, 2017

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

Version 0.8.7.7

Date 2017-9-29

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 (>= 3.3.0),
magrittr (>= 1.5),
ggplot2 (>= 2.1.0),
lubridate (>= 1.6.0),
tidyr (>= 0.7),
dplyr (>= 0.7.2)

Imports tibble (>= 1.3),
rlang (>= 0.1.1),
broom (>= 0.4.2),
car (>= 2.1.3),
grid (>= 3.3.1),
gridExtra (>= 2.0.0),
knitr (>= 1.15.1),
rmarkdown(>= 1.4),
markdown (>= 0.7.7),
pryr (>= 0.1.2),
shiny (>= 1.0.5),
jsonlite (>= 1.0),
shinyAce (>= 0.2.1),
psych (>= 1.6.6),
DT (>= 0.2),
readr (>= 1.1.0),
scales (>= 0.4.0),
curl (>= 2.5),
rstudioapi (>= 0.7),
import (>= 1.1.0),
plotly (>= 4.5.6),
base64enc,
methods

Suggests RSQLite (>= 1.1.2),
DBI (>= 0.6.1),

webshot (>= 0.4.0),
feather (>= 0.3.1),
testthat (>= 1.0.0)

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

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

License AGPL-3 | file LICENSE

LazyData true

RoxygenNote 6.0.1

R topics documented:

`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"))
```

`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 bewteen 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_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_tibble *Exporting as_tibble*

Description

Exporting as_tibble

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 numberic variable return x - mean(x)

changedata *Change data*

Description

Change data

Usage

```
changedata(dataset, vars = c(), var_names = names(vars))
```

Arguments

dataset	Name of the dataframe to change
vars	New variables to add to the data.frame
var_names	Names for the new variables to add to the data.frame

Value

None

choose.dir — *Select a directory. Uses JavaScript on Mac, utils::choose.dir on Windows, and dirname(file.choose()) on Linux*

Description

Select a directory. Uses JavaScript on Mac, utils::choose.dir on Windows, and dirname(file.choose()) on Linux

Usage

```
choose.dir(...)
```

Arguments

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

Value

Path to the directory selected by the user

Examples

```
if (interactive()) {
choose.dir()
}
```

choose.files — *Select files. Uses JavaScript on Mac, utils::choose.files on Windows, and file.choose() on Linux*

Description

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

Usage

```
choose.files(...)
```

Arguments

...	Strings used to determine which file types are available for selection (e.g., "csv" or "pdf")

Value

Vector of paths to files selected by the user

Examples

```
if (interactive()) {
choose.files("pdf", "csv")
}
```

---

ci_label *Labels for confidence intervals*

Description

Labels for confidence intervals

Usage

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

Arguments

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

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)
```

`combinedata` *Combine datasets using dplyr's bind and join functions*

Description

Combine datasets using dplyr's bind and join functions

Usage

```
combinedata(dataset, cmb_dataset, by = "", add = "", type = "inner_join",
  name = "", data_filter = "")
```

Arguments

`dataset`	Dataset name (string). This can be a dataframe in the global environment or an element in an r_data list from Radiant
`cmb_dataset`	Dataset name (string) to combine with 'dataset'. This can be a dataframe in the global environment or an element in an r_data list from Radiant
`by`	Variables used to combine 'dataset' and 'cmb_dataset'
`add`	Variables to add from 'cmb_dataset'
`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
`name`	Name for the combined dataset
`data_filter`	Expression used to filter the dataset. This should be a string (e.g., "price > 10000")

Details

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

Value

If list 'r_data' exists the combined dataset is added as 'name'. Else the combined dataset will be returned as 'name'

Examples

```
avengers %>% combinedata(superheroes, type = "bind_cols")
combinedata("avengers", "superheroes", type = "bind_cols")
avengers %>% combinedata(superheroes, type = "bind_rows")
avengers %>% combinedata(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/smbache/import/issues/4 for a discussion. This function will be depracated when (if) it is included in https://github.com/smbache/import

Examples

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

copy_attr *Copy attributes from on object to another*

Description

Copy attributes from on 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/smbache/import/issues/4 for a discussion. This function will be depracated when (if) it is included in https://github.com/smbache/import

Examples

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

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))
```

describe *Show dataset desription, if available, in html form in Rstudio viewer or default browser*

Description

Show dataset desription, if available, in html form in Rstudio viewer or default browser

Usage

```
describe(name)
```

Arguments

name	Dataset name or a dataframe

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 bundeled 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, funs(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.explore to create the an interactivce table from an explore object

See dtab.pivotr to create the an interactivce table from a pivotr object

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

dtab.character *Create a DT table with bootstrap theme*

Description

Create a DT table with bootstrap theme

Usage

```
## S3 method for class 'character'
dtab(...)
```

Arguments

...	Arguments to pass on to dtab.data.frame

Details

View, search, sort, etc. your data. For styling options see http://rstudio.github.io/DT/functions.html

Examples

```
dtab("mtcars")
```

`dtab.data.frame` *Create a DT table with bootstrap theme*

Description

Create a DT table with bootstrap theme

Usage

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

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 "")
`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)
`na.rm`	Remove rows with missing values (default is FALSE)
`dec`	Number of decimal places to show. Default is no rounding (NULL)
`filter`	Show filter 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
`...`	Additional arguments

Details

View, search, sort, etc. your data. For styling options see http://rstudio.github.io/DT/functions.html

Examples

```
dtab(mtcars)
```

dtab.explore *Make a tabel of summary statistics in DT*

Description

Make a tabel of summary statistics in DT

Usage

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

Arguments

object	Return value from explore
dec	Number of decimals to show
searchCols	Column search and filter. Used to save and restore state
order	Column sorting. Used to save and restore state
pageLength	Page length. Used to save and restore state
...	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 the pivot-table using dplyr

summary.pivotr to print a plain text table

Examples

```
tab <- explore("diamonds", "price:x") %>% dtab
tab <- explore("diamonds", "price", byvar = "cut", fun = c("length", "skew"), top = "byvar") %>%
  dtab
```

`dtab.pivotr` *Make a pivot tabel in DT*

Description

Make a pivot tabel in DT

Usage

```
## S3 method for class 'pivotr'
dtab(object, format = "none", perc = FALSE, dec = 3,
  searchCols = NULL, order = NULL, pageLength = 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. Used to save and restore state
`order`	Column sorting. Used to save and restore state
`pageLength`	Page length. Used to save and restore state
`...`	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

`summary.pivotr` to print a plain text table

Examples

```
pivotr("diamonds", cvars = "cut") %>% dtab
pivotr("diamonds", cvars = c("cut","clarity")) %>% dtab(format = "color_bar")
ret <-  pivotr("diamonds", cvars = c("cut","clarity"), normalize = "total") %>%
   dtab(format = "color_bar", perc = TRUE)
```

empty_level — *Convert categorical variables to factors and deal with empty/missing values (used in pivotr and explore)*

Description

Convert categorical variables to factors and deal with empty/missing values (used in pivotr and explore)

Usage

```
empty_level(x)
```

Arguments

x	Categorical variable used in table

Value

Variable with updated levels

explore — *Explore data*

Description

Explore data

Usage

```
explore(dataset, vars = "", byvar = "", fun = c("mean_rm", "sd_rm"),
  top = "fun", tabfilt = "", tabsort = "", nr = NULL,
  data_filter = "", shiny = FALSE)
```

Arguments

dataset	Dataset name (string). This can be a dataframe in the global environment or an element in an r_data list from Radiant
vars	(Numerical) variables to summaries
byvar	Variable(s) to group data by before summarizing
fun	Functions to use for summarizing
top	The variable (type) to display at the top of the table
tabfilt	Expression used to filter the table. This should be a string (e.g., "Total > 10000")
tabsort	Expression used to sort the table (e.g., "-Total")
nr	Number of rows to display
data_filter	Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
shiny	Logical (TRUE, FALSE) to indicate if the function call originate inside a shiny app

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

```
result <- explore("diamonds", "price:x")
summary(result)
result <- explore("diamonds", c("price","carat"), byvar = "cut", fun = c("n_missing", "skew"))
summary(result)
diamonds %>% explore("price", byvar = "cut", fun = c("length", "n_distinct"))
```

---

factorizer *Convert character to factors as needed*

Description

Convert character to factors as needed

Usage

```
factorizer(dat, safx = 30)
```

Arguments

dat	Data frame
safx	Values to levels ratio

Value

Data frame with factors

filterdata *Filter data with user-specified expression*

Description

Filter data with user-specified expression

Usage

```
filterdata(dat, filt = "")
```

Arguments

dat	Data frame to filter
filt	Filter expression to apply to the specified dataset (e.g., "price > 10000" if dataset is "diamonds")

Value

Filtered data frame

find_dropbox *Find a user's Dropbox folder*

Description

Find a user's Dropbox folder

Usage

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

Arguments

account	If multiple accounts exist specifies the one to use. By default, the first account listed is used

Value

Path to Dropbox account

find_gdrive *Find a user's Google Drive folder*

Description

Find a user's Google Drive folder

Usage

```
find_gdrive()
```

Value

Path to Google Drive folder

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 generate summaries

dtab.explore to create the DT table

Examples

```
result <- explore("diamonds", "price:x", top = "var")
result <- explore("diamonds", "price", byvar = "cut", fun = c("length", "skew"), top = "byvar")
```

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

Description

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

Usage

```
formatdf(tbl, dec = 3, perc = FALSE, mark = "")
```

Arguments

tbl	Data.frame
dec	Number of decimal places
perc	Display numbers as percentages (TRUE or FALSE)
mark	Thousand separator

Value

Data.frame for printing

Examples

```
data.frame(x = c("a","b"), y = c(1L, 2L), z = c(-0.0005, 3)) %>%
  formatdf(dec = 3)
data.frame(x = c(1L, 2L), y = c(0.05, 0.8)) %>%
  formatdf(dec = 2, perc = TRUE)
```

formatnr — *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

```
formatnr(x, sym = "", dec = 2, perc = FALSE, mark = ",")
```

Arguments

x	Number or vector
sym	Symbol to use
dec	Number of decimal places
perc	Display number as a percentage
mark	Thousand separator

Value

Character (vector) in the desired format

Examples

```
formatnr(2000, "$")
formatnr(2000, dec = 4)
formatnr(.05, perc = TRUE)
formatnr(c(.1, .99), perc = TRUE)
formatnr(data.frame(a = c(.1, .99)), perc = TRUE)
formatnr(data.frame(a = 1000), sym = "$", dec = 0)
```

---

getclass *Get variable class*

Description

Get variable class

Usage

```
getclass(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

```
getclass(mtcars)
```

getdata *Get data for analysis functions*

Description

Get data for analysis functions

Usage

```
getdata(dataset, vars = "", filt = "", rows = NULL, na.rm = TRUE)
```

Arguments

dataset	Name of the dataframe
vars	Variables to extract from the dataframe
filt	Filter to apply to the specified dataset. For example "price > 10000" if dataset is "diamonds" (default is "")
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)
na.rm	Remove rows with missing values (default is TRUE)

Value

Data.frame with specified columns and rows

getsummary *Create data.frame summary*

Description

Create data.frame summary

Usage

```
getsummary(dat, dc = getclass(dat))
```

Arguments

dat	Data.frame
dc	Class for each variable

Details

Used in Radiant's Data > Transform tab

ggplotly *Exporting the ggplotly function from the plotly package*

Description

Exporting the ggplotly function from the plotly package

glance *Exporting glance from broom*

Description

Exporting glance from broom

indexr *Find index corrected for missing values and filters*

Description

Find index corrected for missing values and filters

Usage

```
indexr(dataset, vars = "", filt = "", cmd = "")
```

Arguments

dataset	Dataset name
vars	Variables to select
filt	Data filter
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 character variable defined*

Description

Is a character variable defined

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 NULL or an empty string

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_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_string` *Is input a string?*

Description

Is input a string?

Usage

```
is_string(x)
```

Arguments

x	Input

Details

Is input a string

Value

TRUE if string, else FALSE

Examples

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

---

iterms *Create a vector of interaction terms*

Description

Create a vector of interaction terms

Usage

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

Arguments

vars	Variables lables to use
nway	2-way (2) or 3-way (3) interactions labels to create
sep	Separator between variable names (default is :)

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 = ".")
```

---

knit_print *Exporting knit_print from knitr*

Description

Exporting knit_print from knitr

kurtosi *Exporting the kurtosi function from the psych package*

Description

Exporting the kurtosi function from the psych package

level_list *Generate list of levels and unique values*

Description

Generate list of levels and unique values

Usage

```
level_list(dat, ...)
```

Arguments

dat	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))
```

`loadcsv` *Load a csv file with read.csv and read_csv*

Description

Load a csv file with read.csv and read_csv

Usage

```
loadcsv(fn, .csv = FALSE, header = TRUE, sep = ",", dec = ".",
  n_max = Inf, saf = TRUE, safx = 20)
```

Arguments

`fn`	File name string
`.csv`	Use read.csv instead of read_csv to load file (default is FALSE)
`header`	Header in file (TRUE, FALSE)
`sep`	Use , (default) or ; or \t
`dec`	Decimal symbol. Use . (default) or ,
`n_max`	Maximum number of rows to read
`saf`	Convert character variables to factors if (1) there are less than 100 distinct values (2) there are X (see safx) more values than levels
`safx`	Values to levels ratio

Value

Data frame with (some) variables converted to factors

`loadcsv_url` *Load a csv file with from a url*

Description

Load a csv file with from a url

Usage

```
loadcsv_url(csv_url, header = TRUE, sep = ",", dec = ".", n_max = Inf,
  saf = TRUE, safx = 20)
```

Arguments

csv_url	URL for the csv file
header	Header in file (TRUE, FALSE)
sep	Use , (default) or ; or \t
dec	Decimal symbol. Use . (default) or ,
n_max	Maximum number of rows to read
saf	Convert character variables to factors if (1) there are less than 100 distinct values (2) there are X (see safx) more values than levels
safx	Values to levels ratio

Value

Data frame with (some) variables converted to factors

---

loadr *Load an rds, rda, or csv file and add it to the radiant data list (r_data) if available*

Description

Load an rds, rda, or csv file and add it to the radiant data list (r_data) if available

Usage

```
loadr(file, objname = "", rlist = TRUE)
```

Arguments

file	File name and path as a string. Extension must be either rds, rda, or csv
objname	Name to use for the data frame. Defaults to the file name
rlist	If TRUE, uses "r_data" list to store the data.frame. If FALSE, loads data.frame into calling environment

Value

Data frame in r_data or in the calling enviroment

loadrda_url *Load an rda file from a url*

Description

Load an rda file from a url

Usage

```
loadrda_url(rda_url)
```

Arguments

rda_url	URL for the rda file

Value

Data frame

make_funs *Make a list of functions-as-formulas to pass to dplyr*

Description

Make a list of functions-as-formulas to pass to dplyr

Usage

```
make_funs(x)
```

Arguments

x	List of functions as strings

Value

List of functions to pass to dplyr in formula form

Examples

```
make_funs(c("mean", "sum_rm"))
```

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 = 100, seed = 1234)
```

Arguments

n	Number (or fraction) of observations to label as training
nr	Number of rows in the dataset
seed	Random seed

Value

0/1 variables for filtering

Examples

```
make_train(.5, 10)
```

max_rm *Max with na.rm = TRUE*

Description

Max with na.rm = TRUE

Usage

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

Arguments

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

Value

Maximum value

Examples

```
max_rm(runif (100))
```

mean_rm *Mean with na.rm = TRUE*

Description

Mean with na.rm = TRUE

Usage

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

Arguments

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

Value

Mean value

Examples

```
mean_rm(runif (100))
```

median_rm *Median with na.rm = TRUE*

Description

Median with na.rm = TRUE

Usage

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

Arguments

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

Value

Median value

Examples

```
median_rm(runif (100))
```

min_rm *Min with na.rm = TRUE*

Description

Min with na.rm = TRUE

Usage

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

Arguments

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

Value

Minimum value

Examples

```
min_rm(runif (100))
```

mode_rm *Mode with na.rm = TRUE*

Description

Mode with na.rm = TRUE

Usage

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

Arguments

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

Value

Mode value

Examples

```
mode_rm(diamonds$cut)
```

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 tranformed variables to a data frame (NSE)*

Description

Add tranformed variables to a data frame (NSE)

Usage

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

Arguments

.tbl	Data frame to add transformed variables to
.funs	Function(s) to apply (e.g., funs(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

Value

number of missing values

Examples

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

p05 *5th percentile*

Description

5th percentile

Usage

```
p05(x, na.rm = TRUE)
```

Arguments

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

Value

5th percentile

Examples

```
p05(rnorm(100))
```

p10 *10th percentile*

Description

10th percentile

Usage

```
p10(x, na.rm = TRUE)
```

Arguments

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

Value

10th percentile

Examples

```
p10(rnorm(100))
```

p25 *25th percentile*

Description

25th percentile

Usage

```
p25(x, na.rm = TRUE)
```

Arguments

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

Value

25th percentile

Examples

```
p25(rnorm(100))
```

p75 *75th percentile*

Description

75th percentile

Usage

```
p75(x, na.rm = TRUE)
```

Arguments

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

Value

75th percentile

Examples

```
p75(rnorm(100))
```

p90 *90th percentile*

Description

90th percentile

Usage

```
p90(x, na.rm = TRUE)
```

Arguments

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

Value

90th percentile

Examples

```
p90(rnorm(100))
```

p95 *95th percentile*

Description

95th percentile

Usage

```
p95(x, na.rm = TRUE)
```

Arguments

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

Value

95th percentile

Examples

```
p95(rnorm(100))
```

`pivotr` *Create a pivot table using dplyr*

Description

Create a pivot table using dplyr

Usage

```
pivotr(dataset, cvars = "", nvar = "None", fun = "mean_rm",
  normalize = "None", tabfilt = "", tabsort = "", nr = NULL,
  data_filter = "", shiny = FALSE)
```

Arguments

`dataset`	Name of the dataframe to change
`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. This should be a string (e.g., "Total > 10000")
`tabsort`	Expression used to sort the table (e.g., "-Total")
`nr`	Number of rows to display
`data_filter`	Expression used to filter the dataset. This should be a string (e.g., "price > 10000")
`shiny`	Logical (TRUE, FALSE) to indicate if the function call originate inside a shiny app

Details

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

Examples

```
result <- pivotr("diamonds", cvars = "cut")$tab
result <- pivotr("diamonds", cvars = c("cut","clarity","color"))$tab
result <- pivotr("diamonds", cvars = "cut:clarity", nvar = "price")$tab
result <- pivotr("diamonds", cvars = "cut", nvar = "price")$tab
result <- pivotr("diamonds", cvars = "cut", normalize = "total")$tab
```

plot.character *Don't try to plot strings*

Description

Don't try to plot strings

Usage

```
## S3 method for class 'character'
plot(x, ...)
```

Arguments

x	A character returned from a function
...	Any additional arguments

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", ...)
```

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")
...	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
```

`print.gtable` *Print/draw method for grobs produced by gridExtra*

Description

Print/draw method for grobs produced by gridExtra

Usage

```
## S3 method for class 'gtable'
print(x, ...)
```

Arguments

`x`	a gtable object
`...`	further arguments passed to or from other methods

Details

Print method for ggplot grobs created using grid.arrange. Code is based on https://github.com/baptiste/gridextra/blob/master/inst/testing/shiny.R

Value

A 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 http://stat545-ubc.github.io/bit001_dplyr-cheatsheet.html. The dataset is used to illustrate data merging / joining. Description provided in attr(publishers,"description")

---

radiant.data *radiant.data*

Description

radiant.data

Launch Radiant in the default browser

Usage

```
radiant.data()
```

Details

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

radiant.data-deprecated

Deprecated function(s) in the radiant.data package

Description

These functions are provided for compatibility with previous versions of radiant. They will eventually be removed.

Usage

```
mutate_each(...)
```

Arguments

...	Parameters to be passed to the updated functions

Details

mutate_each is now a synonym for mutate_ext, mutate_at, or mutate_all
dfprint is now a synonym for formatdf
nrprint is now a synonym for formatnr
varp_rm is now a synonym for varpop
sdp_rm is now a synonym for sdpop

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 relable 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 in the datasetlist in Radiant*

Description

Register a data.frame in the datasetlist in Radiant

Usage

```
register(new = "", org = "", descr = "", envir = parent.frame(), ...)
```

Arguments

new	Name of the new dataset
org	Name of the original data
descr	Dataset description
envir	Environment to assign 'new' dataset (optional). Used if 'new' is specified but an r_data list is not available
...	further arguments passed to or from other methods

Details

Store data frame in Radiant r_data list if available

---

render *Method to render objects (i.e., htmlwidgets and rmarkdown files)*

Description

Method to render objects (i.e., htmlwidgets and rmarkdown files)

Usage

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

Arguments

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

render.character *Method to render rmarkdown documents*

Description

Method to render rmarkdown documents

Usage

```
## S3 method for class 'character'
render(object, ...)
```

Arguments

object	File path to an R-markdown file
...	Additional arguments passed on to rmarkdown::render

render.datatables *Method to render DT tabels*

Description

Method to render DT tabels

Usage

```
## S3 method for class 'datatables'
render(object, ...)
```

Arguments

object	DT table
...	Additional arguments

render.plotly *Method to render plotly plots*

Description

Method to render plotly plots

Usage

```
## S3 method for class 'plotly'
render(object, ...)
```

Arguments

object	ggplotly object
...	Additional arguments

render.shiny.render.function *Method to avoid re-rendering a shiny.render.function*

Description

Method to avoid re-rendering a shiny.render.function

Usage

```
## S3 method for class 'shiny.render.function'
render(object, ...)
```

Arguments

object	Shiny render function
...	Additional arguments

rounddf *Round double in a data.frame to a specified number of decimal places*

Description

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

Usage

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

Arguments

tbl	Data frame
dec	Number of decimal places

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)) %>%
  rounddf(dec = 3)
```

rownames_to_column *Exporting rownames_to_column from tibble*

Description

Exporting rownames_to_column from tibble

saver *Save data.frame as an rda or rds file from Radiant*

Description

Save data.frame as an rda or rds file from Radiant

Usage

```
saver(objname, file)
```

Arguments

objname	Name of a data.frame or a data.frame
file	File name and path as a string. Extension must be either rda or rds

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)))
```

sd_rm *Standard deviation with na.rm = TRUE*

Description

Standard deviation with na.rm = TRUE

Usage

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

Arguments

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

Value

Standard deviation

Examples

```
sd_rm(rnorm(100))
```

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 *Search for a string in all columns of a data.frame*

Description

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

Usage

```
Search(pattern, df, ignore.case = TRUE, fixed = FALSE)
```

Arguments

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

Details

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

See Also

See grepl for a more detailed description of the function arguments

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("desc", "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 '***' to a data.frame (from broom's 'tidy' function) based on p.values*

Description

Add stars '***' to a data.frame (from broom's 'tidy' function) based on p.values

Usage

```
sig_stars(pval)
```

Arguments

pval	Vector of p-values

Details

Add stars to output from broom's 'tidy' function

Value

A vector of stars

Examples

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

`skew` *Exporting the skew function from the psych package*

Description

Exporting the skew function from the psych package

`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

Adapted from http://www.onthelambda.com/2014/09/17/fun-with-rprofile-and-customizing-r-startup/

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

Adapted from http://www.onthelambda.com/2014/09/17/fun-with-rprofile-and-customizing-r-startup/

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 numberic variable return center(x) / mean(x)

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

Description

Method to store variables in a dataset in Radiant

Usage

```
store(object, ...)
```

Arguments

object	Object of relevant class that has required information to store
...	Additional arguments

store.character *Method for error messages that a user tries to store*

Description

Method for error messages that a user tries to store

Usage

```
## S3 method for class 'character'
store(object, ...)
```

Arguments

object	Object of type character
...	Additional arguments

store.data.frame *Store method for the Data > View tab*

Description

Store method for the Data > View tab

Usage

```
## S3 method for class 'data.frame'
store(object, new = "", org = "",
  envir = parent.frame(), ...)
```

Arguments

object	Filtered data frame from the Data > View tab
new	Name of the new dataset
org	Name of the original data
envir	Environment to assign 'new' dataset (optional). Used if 'new' is specified but an r_data list is not available
...	further arguments passed to or from other methods

Details

Store data frame in Radiant r_data list if available

store.explore *Store method for the explore function*

Description

Store method for the explore function

Usage

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

Arguments

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

Details

Add the summarized data to the r_data list in Radiant or return it. See https://radiant-rstats.github.io/docs/data/explore.html for an example in Radiant

See Also

explore to generate summaries

store.pivotr *Store method for the pivort function*

Description

Store method for the pivort function

Usage

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

Arguments

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

Details

Add the summarized data to the r_data list in Radiant or return it. See https://radiant-rstats.github.io/docs/data/pivotr.html for an example in Radiant

See Also

pivotr to generate summaries

`subplot` *Exporting the subplot function from the plotly package*

Description

Exporting the subplot function from the plotly package

`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("length", "skew"))
summary(result)
diamonds %>% explore("price:x") %>% summary
diamonds %>% explore("price", byvar = "cut", fun = c("length", "skew")) %>% 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 = "-n") %>% summary
pivotr("diamonds", cvars = "cut", tabsort = "desc(n)") %>% summary
pivotr("diamonds", cvars = "cut", tabfilt = "n > 700") %>% summary
pivotr("diamonds", cvars = "cut:clarity", nvar = "price") %>% summary
```

sum_rm *Sum with na.rm = TRUE*

Description

Sum with na.rm = TRUE

Usage

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

Arguments

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

Value

Sum of input values

Examples

```
sum_rm(1:200)
```

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 http://stat545-ubc.github.io/bit001_dplyr-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(dat, freq = tail(colnames(dat), 1))
```

Arguments

dat	Data.frame
freq	Column name with frequency information

Examples

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

tibble *Exporting tibble*

Description

Exporting tibble

tidy *Exporting tidy from broom*

Description

Exporting tidy from broom

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")

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)))
```

var_rm *Variance with na.rm = TRUE*

Description

Variance with na.rm = TRUE

Usage

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

Arguments

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

Value

Variance

Examples

```
var_rm(rnorm(100))
```

viewdata *View data in a shiny-app*

Description

View data in a shiny-app

Usage

```
viewdata(dataset, vars = "", filt = "", rows = NULL, na.rm = FALSE)
```

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. For example "price > 10000" if dataset is "diamonds" (default is "")
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)
na.rm	Remove rows with missing values (default is FALSE)

Details

View, search, sort, etc. your data

Examples

```
if (interactive()) {
viewdata(mtcars)
viewdata("mtcars")
mtcars %>% viewdata
}
```

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

Description

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

Usage

```
visualize(dataset, xvar, yvar = "", comby = FALSE, combx = FALSE,
  type = "dist", 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, ylim = "none", data_filter = "",
  shiny = FALSE, custom = FALSE)
```

Arguments

dataset	Dataset name (string). This can be a dataframe in the global environment or an element in an r_data list from Radiant
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')
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)
ylim	Set limit for y-axis
data_filter	Expression used to filter the dataset. This should be a string (e.g., "price > 10000")
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 opion can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and http://docs.ggplot2.org/ for options.

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(dataset = "diamonds", yvar = "price", xvar = c("cut","clarity"),
  type = "bar", fun = "median")
visualize(dataset = "diamonds", yvar = "price", xvar = c("cut","clarity"),
  type = "line", fun = "max")
visualize(dataset = "diamonds", yvar = "price", xvar = "carat", type = "scatter",
          size = "table", custom = TRUE) + scale_size(range=c(1,10), guide = "none")
visualize(dataset = "diamonds", yvar = "price", xvar = "carat", type = "scatter", custom = TRUE) +
  labs(title = "A scatterplot", x = "price in $")
visualize(dataset = "diamonds", xvar = "price:carat", custom = TRUE) %>%
  gridExtra::grid.arrange(grobs = ., top = "Histograms", ncol = 2)
visualize(dataset = "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 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

Calculated a weighted standard deviation

which.pmax *Returns the index of the (parallel) maxima of the input values*

Description

Returns the index of the (parallel) maxima of the input values

Usage

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

Arguments

...	Numeric or character vectors of the same length

Value

Vector of rankings

Examples

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

which.pmin *Returns the index of the (parallel) minima of the input values*

Description

Returns the index of the (parallel) minima of the input values

Usage

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

Arguments

...	Numeric or character vectors of the same length

Value

Vector of rankings

Examples

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

xtile *Create quantiles*

Description

Create quantiles

Usage

```
xtile(x, n, rev = FALSE)
```

Arguments

x	Numeric variable
n	number of bins to create
rev	Reverse the order of the xtiles

Details

Approach used produces results most similar to Stata

Examples

```
xtile(1:10,5)
xtile(1:10,5, rev = TRUE)
```

Index