# Package 'radiant.data'

May 6, 2018

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

**Version** 0.9.3.4

**Date** 2018-4-6

**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 (>= 3.4.0),
magrittr (>= 1.5),
ggplot2 (>= 2.2.1),
lubridate (>= 1.7.4),
tidyr (>= 0.8.0),
dplyr (>= 0.7.4)

**Imports** tibble (>= 1.4.2),
rlang (>= 0.2.0),
broom (>= 0.4.3),
car (>= 3.0-0),
grid (>= 3.3.1),
gridExtra (>= 2.0.0),
knitr (>= 1.20),
markdown (>= 0.8),
rmarkdown(>= 1.9),
pryr (>= 0.1.2),
shiny (>= 1.0.5),
jsonlite (>= 1.0),
shinyAce (>= 0.3.0.1),
psych (>= 1.8.3.3),
DT (>= 0.4),
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),
feather (>= 0.3.1),
glue (>= 1.2.0),
base64enc,

methods

**Suggests** DBI (>= 0.7),
RSQLite (>= 2.0),
odbc (>= 1.1.4),
webshot (>= 0.5.0),
testthat (>= 2.0.0)

**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

**RoxygenNote** 6.0.1

**VignetteBuilder** knitr

# 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 from tibble*

### Description

Exporting as_tibble from tibble

### Details

See as_tibble in the tibble package for more details

## 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)

---

`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

```
## Not run:
choose_dir()

## End(Not run)
```

## 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

```
## 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)
```

---

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

---

## Description

Combine datasets using dplyr's bind and join functions

## Usage

```
combinedata(x, y, by = "", add = "", type = "inner_join",
  data_filter = "", ...)
```

### 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") |
| ... | 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

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 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/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(dataset)
```

## Arguments

| | |
|---|---|
| dataset | Dataset |

## 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.data.frame to create an interactive table from a data.frame

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

## 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,
  nr = NULL, na.rm = FALSE, dec = 3, perc = "", 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) |
| nr | Number of rows of data to include in the table |
| 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 |
| ... | Additional arguments |

## Details

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

## Examples

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

## End(Not run)
```

---

# dtab.explore *Make a table of summary statistics in DT*

---

## Description

Make a table 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("n_obs", "skew"), top = "byvar") %>%
  dtab()
```

---

# dtab.pivotr *Make a pivot table in DT*

---

## Description

Make a pivot table 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

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 (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", "sd"),
  top = "fun", tabfilt = "", tabsort = "", nr = NULL,
  data_filter = "", shiny = FALSE)
```

### Arguments

| | |
|---|---|
| dataset | Dataset to explore |
| 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("n_obs", "n_distinct"))
```

---

## filterdata *Filter data with user-specified expression*

---

### Description

Filter data with user-specified expression

### Usage

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

### Arguments

| | |
|---|---|
| dataset | Data frame to filter |
| filt | Filter expression to apply to the specified dataset (e.g., "price > 10000" if dataset is "diamonds") |
| drop | Drop unused factor levels after filtering (default is TRUE) |

### 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

## find_project *Find the rstudio project directory*

### Description

Find the rstudio project directory

### Usage

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

### Arguments

| | |
|---|---|
| mess | Show or hide messages (default mess = TRUE) |

### Value

Path to rstudio project directory

## fixMS *Replace Windows smart quotes etc.*

### Description

Replace Windows smart quotes etc.

### Usage

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

### Arguments

| | |
|---|---|
| text | Text to be parsed |
| all | Should all non-ascii characters be removed (default = FALSE) |

## fix_names — *Make column names that are valid in R*

### Description

Make column names that are valid in R

### Usage

```
fix_names(x)
```

### Arguments

| | |
|---|---|
| x | Data.frame or vector of column names |

### Details

Removes symbols, trailing and leading spaces and converts to valid R column names

## 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("n_obs", "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 = NULL, perc = FALSE, mark = "", ...)
```

### Arguments

| | |
|---|---|
| tbl | Data.frame |
| dec | Number of decimals to show |
| perc | Display numbers as percentages (TRUE or FALSE) |
| mark | Thousand separator |
| ... | Additional arguments for formatnr |

### Value

Data.frame for printing

### Examples

```
data.frame(x = c("a", "b"), y = c(1L, 2L), z = c(-0.0005, 3)) %>%
  formatdf(dec = 4)
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 = ",", 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

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

---

## 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 | Dataset or name of the data.frame |
| vars | Variables to extract from the data.frame |
| 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(dataset, dc = getclass(dataset))
```

### Arguments

| | |
|---|---|
| dataset | Data.frame |
| dc | Class for each variable |

### 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)

## glance — *Exporting glance from broom*

### Description

Exporting glance from broom

### Details

See glance in the broom package for more details

## glue — *Exporting glue from glue*

### Description

Exporting glue from glue

### Details

See glue in the glue] package for more details

## 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 |
| 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_empty(c("", "something"))
is_empty(c(NA, 1:100))
is_empty(mtcars)
```

## 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_numeric *Is input numeric (and not a date type)?*

---

### Description

Is input numeric (and not a date type)?

### Usage

```
is_numeric(x)
```

### Arguments

| | |
|---|---|
| x | Input |

### Value

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

---

## 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*

---

### 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

### Details

See knit_print in the knitr package for more details

## kurtosi *Exporting kurtosi from psych*

### Description

Exporting kurtosi from psych

### Details

See kurtosi in the psych package for more details

## launch *Launch radiant apps*

### Description

Launch radiant apps

### Usage

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

### 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") |

### 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

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

## 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)
```

## 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 |
| ... | 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))
```

## p025 *2.5th percentile*

### Description

2.5th percentile

### Usage

```
p025(x, na.rm = TRUE)
```

### Arguments

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

### Value

2.5th percentile

### Examples

```
p025(rnorm(100))
```

## 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))
```

## p975 *97.5th percentile*

### Description

97.5th percentile

### Usage

```
p975(x, na.rm = TRUE)
```

### Arguments

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

### Value

97.5th percentile

### Examples

```
p975(rnorm(100))
```

## parse_path *Parse path into useful components (used by read_files function)*

### Description

Parse path into useful components (used by read_files function)

### Usage

```
parse_path(path, chr = "\"", pdir = getOption("radiant.project_dir", ""))
```

### Arguments

| | |
|---|---|
| path | Path to be parsed |
| chr | Character to wrap around path for display |
| pdir | Project directory if available |

pivotr *Create a pivot table using dplyr*

## Description

Create a pivot table using dplyr

## Usage

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

## 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. 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

```
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.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", 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()
```

---

`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 the radiant.data app in the default web browser

## Usage

```
radiant.data()
```

## Examples

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

## 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()
```

### 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()
```

### Examples

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

## End(Not run)
```

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

### Description

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

### Usage

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

### Arguments

| | |
|---|---|
| path | Path to file. If empty, a file browser will be opened |
| 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? |

## 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 or list in Radiant*

### Description

Register a data.frame or list in Radiant

### Usage

```
register(new, org = "", descr = "", env)
```

### 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 |
| env | Environment to assign data to |

## 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 tables*

### Description

Method to render DT tables

### 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 | plotly 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 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)) %>%
  rounddf(dec = 2)
```

## rownames_to_column *Exporting rownames_to_column from tibble*

### Description

Exporting rownames_to_column from tibble

### Details

See rownames in the tibble package for more details

## save_clip *Save data.frame or tibble to clipboard on Windows or macOS*

### Description

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

### Usage

```
save_clip(dataset)
```

### Arguments

| | |
|---|---|
| dataset | Dataset to push to clipboard |

### Details

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

## 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 *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, dataset, ignore.case = TRUE, fixed = FALSE)
```

### Arguments

| | |
|---|---|
| pattern | String to match |
| dataset | 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 skew from psych*

### Description

Exporting skew from psych

### Details

See skew in the psych package for more details

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

### Arguments

| | |
|---|---|
| dataset | Dataset |
| object | Object of relevant class that has information to be stored |
| ... | 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(dataset = NULL, object, ...)
```

### Arguments

| | |
|---|---|
| dataset | Dataset |
| object | Object of type character |
| ... | 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)
diamonds %>% explore("price:x") %>% summary()
diamonds %>% explore("price", byvar = "cut", fun = c("n_obs", "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_obs") %>% summary()
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 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(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()
```

## tibble *Exporting tibble from tibble*

### Description

Exporting tibble from tibble

### Details

See tibble in the tibble package for more details

## tidy *Exporting tidy from broom*

### Description

Exporting tidy from broom

### Details

See tidy in the broom package for more details

## 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")

## toFct *Convert character to factors as needed*

### Description

Convert character to factors as needed

### Usage

```
toFct(dataset, safx = 30)
```

### Arguments

| | |
|---|---|
| dataset | Data frame |
| safx | Values to levels ratio |

### Value

Data frame with factors

## 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)))
```

## viewdata *View data in a shiny-app*

### Description

View data in a shiny-app

### Usage

```
viewdata(dataset, vars = "", filt = "", rows = NULL, na.rm = FALSE,
  dec = 3)
```

### 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) |
| dec | Number of decimals to show |

### Details

View, search, sort, etc. your data

### Examples

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

## End(Not run)
```

## 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 = 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, xlim = NULL, ylim = NULL, data_filter = "",
  shiny = FALSE, custom = FALSE)
```

## 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) |
| xlim | Set limit for y-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") |
| 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(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) %>%
  gridExtra::grid.arrange(grobs = ., top = "Histograms", ncol = 2)
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

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.pmax(mtcars)
```

## 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)
which.pmin(mtcars)
```

## write_feather *Workaround to add description using feather::write_feather*

### Description

Workaround to add description using feather::write_feather

### Usage

```
write_feather(x, path, description = attr(x, "description"))
```

### Arguments

| | |
|---|---|
| x | A data frame to write to disk |
| path | Path to feather file |
| description | Data description |

`xtile` *Create quantiles*

## Description

Create quantiles

## Usage

```
xtile(x, n = 5, 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