

# Package ‘radiant.data’

August 1, 2016

**Title** Business Analytics using R and Shiny

**Version** 0.5.10

**Date** 2016-7-31

**Description** A platform-independent browser-based interface for business analytics in R, based on the Shiny package.

**Depends** R (>= 3.3.0),  
magrittr (>= 1.5),  
ggplot2 (>= 2.0.0),  
lubridate (>= 1.5.0),  
tidyr (>= 0.4.1),  
dplyr (>= 0.5)

**Imports** tibble (>= 1.1),  
broom (>= 0.4.0),  
car (>= 2.1.1),  
gridExtra (>= 2.0.0),  
knitr (>= 1.13),  
rmarkdown (>= 0.9.5),  
markdown (>= 0.7.7),  
pryr (>= 0.1.2),  
shiny (>= 0.13.2),  
jsonlite (>= 0.9.17),  
shinyAce (>= 0.2.1),  
psych (>= 1.5.8),  
DT (>= 0.1.55),  
readr (>= 0.2.2),  
scales (>= 0.4.0),  
curl (>= 0.9.4),  
rstudioapi (>= 0.5),  
import (>= 1.1.0),  
base64enc,  
methods

**Suggests** testthat (>= 1.0.0),  
covr (>= 1.2.0)

**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** 5.0.1

## R topics documented:

add_class . . . . .	4
as_character . . . . .	5
as_data_frame . . . . .	5
as_distance . . . . .	5
as_dmy . . . . .	6
as_dmy_hm . . . . .	6
as_dmy_hms . . . . .	7
as_duration . . . . .	7
as_factor . . . . .	8
as_hm . . . . .	8
as_hms . . . . .	9
as_integer . . . . .	9
as_mdy . . . . .	10
as_mdy_hm . . . . .	11
as_mdy_hms . . . . .	11
as_numeric . . . . .	12
as_tibble . . . . .	12
as_ymd . . . . .	12
as_ymd_hm . . . . .	13
as_ymd_hms . . . . .	13
avengers . . . . .	14
center . . . . .	14
changedata . . . . .	15
ci_label . . . . .	15
ci_perc . . . . .	16
combinedata . . . . .	16
copy_all . . . . .	17
copy_from . . . . .	18
cv . . . . .	19
data_frame . . . . .	19
describe . . . . .	19
diamonds . . . . .	20
does_vary . . . . .	20
explore . . . . .	21
factorizer . . . . .	22
filterdata . . . . .	22
find_dropbox . . . . .	23
flip . . . . .	23
formatdf . . . . .	24
formatnr . . . . .	24
getclass . . . . .	25
getdata . . . . .	26
getsummary . . . . .	26
glance . . . . .	27
indexr . . . . .	27
install_webshot . . . . .	27

inverse . . . . .	28
is_empty . . . . .	28
is_not . . . . .	29
is_string . . . . .	29
iterms . . . . .	30
kurtosi . . . . .	30
level_list . . . . .	31
ln . . . . .	31
loadcsv . . . . .	32
loadcsv_url . . . . .	32
loadr . . . . .	33
loadrda_url . . . . .	33
make_dt . . . . .	34
make_expl . . . . .	35
make_funs . . . . .	36
make_train . . . . .	36
max_rm . . . . .	37
mean_rm . . . . .	37
median_rm . . . . .	38
min_rm . . . . .	38
mode_rm . . . . .	39
mutate_each . . . . .	39
normalize . . . . .	40
n_missing . . . . .	40
p05 . . . . .	41
p10 . . . . .	41
p25 . . . . .	42
p75 . . . . .	42
p90 . . . . .	43
p95 . . . . .	43
pivotr . . . . .	44
plot.character . . . . .	45
plot.pivotr . . . . .	45
print.gtable . . . . .	46
publishers . . . . .	46
radiant.data . . . . .	47
radiant.data-deprecated . . . . .	47
render . . . . .	48
render.datatables . . . . .	48
rounddf . . . . .	48
rownames_to_column . . . . .	49
saver . . . . .	49
sdp_rm . . . . .	50
sd_rm . . . . .	50
serr . . . . .	51
set_attr . . . . .	51
show_duplicated . . . . .	52
sig_stars . . . . .	52
skew . . . . .	53
square . . . . .	53
sshh . . . . .	54
sshhr . . . . .	54

standardize . . . . .	55
store . . . . .	55
summary.explore . . . . .	56
summary.pivotr . . . . .	56
sum_rm . . . . .	57
superheroes . . . . .	58
table2data . . . . .	58
tibble . . . . .	58
tidy . . . . .	59
titanic . . . . .	59
varp_rm . . . . .	59
var_rm . . . . .	60
viewdata . . . . .	60
visualize . . . . .	61
weighted.sd . . . . .	62
which.pmax . . . . .	63
which.pmin . . . . .	63
xtile . . . . .	64
<b>Index</b>	<b>65</b>

---

add_class	<i>Convenience function to add a class</i>
-----------	--

---

**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	<i>Wrapper for as.character</i>
--------------	---------------------------------

---

**Description**

Wrapper for as.character

**Usage**

```
as_character(x)
```

**Arguments**

x                      Input vector

---

as_data_frame	<i>Exporting as_data_frame</i>
---------------	--------------------------------

---

**Description**

Exporting as\_data\_frame

---

as_distance	<i>Distance in kilometers or miles between two locations based on lat-long Function based on <a href="http://www.movable-type.co.uk/scripts/latlong.html">http://www.movable-type.co.uk/scripts/latlong.html</a>. Uses the haversine formula</i>
-------------	--

---

**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	<i>Convert input in day-month-year format to date</i>
--------	---

---

**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	<i>Convert input in day-month-year-hour-minute format to date-time</i>
-----------	--

---

**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	<i>Convert input in day-month-year-hour-minute-second format to date-time</i>
------------	---

---

**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	<i>Wrapper for lubridate's as.duration function. Result converted to numeric</i>
-------------	--

---

**Description**

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

**Usage**

```
as_duration(x)
```

**Arguments**

x	Time difference
---	-----------------

---

as_factor	<i>Wrapper for as.factor</i>
-----------	------------------------------

---

**Description**

Wrapper for as.factor

**Usage**

```
as_factor(x)
```

**Arguments**

x	Input vector
---	--------------

---

as_hm	<i>Convert input in hour-minute format to time</i>
-------	--

---

**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(5:10 %>% as.factor)
as.integer(5:10 %>% as.factor)
```

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	<i>Convert input in month-day-year-hour-minute format to date-time</i>
-----------	--

---

**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	<i>Convert input in month-day-year-hour-minute-second format to date-time</i>
------------	---

---

**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	<i>Convert variable to numeric avoiding potential issues with factors</i>
------------	---

---

**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(5:10 %>% as.factor)
as.numeric(5:10 %>% as.factor)
as_numeric(c("1", "2"))
```

---

as_tibble	<i>Exporting as_tibble</i>
-----------	----------------------------

---

**Description**

Exporting as\_tibble

---

as_ymd	<i>Convert input in year-month-day format to date</i>
--------	---

---

**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	<i>Avengers</i>
----------	-----------------

---

**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	<i>Center</i>
--------	---------------

---

**Description**

Center

**Usage**

```
center(x)
```

**Arguments**

x                      Input variable

**Value**

If x is a numeric variable return  $x - \text{mean}(x)$

---

changedata	<i>Change data</i>
------------	--------------------

---

**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

---

ci_label	<i>Labels for confidence intervals</i>
----------	--

---

**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	<i>Values at confidence levels</i>
---------	------------------------------------

---

**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	<i>Combine datasets using dplyr's bind and join functions</i>
-------------	---

---

**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 <code>r_data</code> 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 <code>r_data</code> list from Radiant
by	Variables used to combine 'dataset' and 'cmb_dataset'
add	Variables to add from 'cmb_dataset'





**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 deprecated when (if) it is included in <https://github.com/smbache/import>

**Examples**

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

---

copy_from	<i>Source for package functions</i>
-----------	-------------------------------------

---

**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 deprecated when (if) it is included in <https://github.com/smbache/import>

**Examples**

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

---

cv	<i>Coefficient of variation</i>
----	---------------------------------

---

**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))
```

---

data_frame	<i>Exporting data_frame</i>
------------	-----------------------------

---

**Description**

Exporting data\_frame

---

describe	<i>Show dataset desription, if available, in html form in Rstudio viewer or default browser</i>
----------	---

---

**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 bundled with ggplot2. Description provided in `attr(diamonds,"description")`

---

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

---

**Description**

Does a vector have non-zero variability?

**Usage**

```
does_vary(x)
```

**Arguments**

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

**Value**

Logical. TRUE if there is variability

**Examples**

```
summarise_each(diamonds, funs(does_vary)) %>% as.logical
```

explore

*Explore data***Description**

Explore data

**Usage**

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

**Arguments**

dataset	Dataset name (string). This can be a dataframe in the global environment or an element in an <code>r_data</code> list from Radiant
vars	(Numerical) variables to summaries
byvar	Variable(s) to group data by before summarizing
fun	Functions to use for summarizing
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")
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 <http://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**

[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	<i>Convert character to factors as needed</i>
------------	---

---

**Description**

Convert character to factors as needed

**Usage**

```
factorizer(dat, safx = 20)
```

**Arguments**

dat	Data frame
safx	Values to levels ratio

**Value**

Data frame with factors

---

filterdata	<i>Filter data with user-specified expression</i>
------------	---

---

**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	<i>Find a user's dropbox folder</i>
--------------	-------------------------------------

---

**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

---

flip	<i>Flip the DT table to put Function, Variable, or Group by on top</i>
------	--

---

**Description**

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

**Usage**

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

**Arguments**

expl	Return value from <a href="#">explore</a>
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 <http://radiant-rstats.github.io/docs/data/explore.html> for an example in Radiant

**See Also**

[explore](#) to generate summaries  
[make\\_expl](#) to create the DT table

**Examples**

```
result <- explore("diamonds", "price:x") %>% flip("var")

result <- explore("diamonds", "price", byvar = "cut", fun = c("length", "skew")) %>%
  flip("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	<i>Get data for analysis functions</i>
---------	--

---

**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	<i>Create data.frame summary</i>
------------	----------------------------------

---

**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

---

glance	<i>Exporting glance from broom</i>
--------	------------------------------------

---

**Description**

Exporting glance from broom

---

indexr	<i>Find index corrected for missing values and filters</i>
--------	--

---

**Description**

Find index corrected for missing values and filters

**Usage**

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

**Arguments**

dataset	Dataset name
vars	Variables to select
filt	Data filter

---

install_webshot	<i>Install webshot and phantomjs</i>
-----------------	--------------------------------------

---

**Description**

Install webshot and phantomjs

**Usage**

```
install_webshot()
```

---

inverse	<i>Calculate inverse of a variable</i>
---------	--

---

**Description**

Calculate inverse of a variable

**Usage**

```
inverse(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

1/x

---

is_empty	<i>Is a character variable defined</i>
----------	--

---

**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 = ".")
```

---

kurtosi

*Exporting the kurtosi function from the psych package*

---

**Description**

Exporting the kurtosi function from the psych package

---

level_list	<i>Generate list of levels and unique values</i>
------------	--

---

**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	<i>Natural log</i>
----	--------------------

---

**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	<i>Load a csv file with read.csv and read_csv</i>
---------	---

---

### Description

Load a csv file with read.csv and read\_csv

### Usage

```
loadcsv(fn, .csv = FALSE, header = TRUE, sep = ",", dec = ".",
  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 ,
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
safox	Values to levels ratio

### Value

Data frame with (some) variables converted to factors

---

loadcsv_url	<i>Load a csv file with from a url</i>
-------------	--

---

### Description

Load a csv file with from a url

### Usage

```
loadcsv_url(csv_url, header = TRUE, sep = ",", dec = ".", saf = TRUE,
  safox = 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 ,
saf	Convert character variables to factors if (1) there are less than 100 distinct values (2) there are X (see safox) more values than levels
safox	Values to levels ratio



**Value**

Data frame with (some) variables converted to factors

---

loadr	<i>Load an rda or rds file and add it to the radiant data list (r_data) if available</i>
-------	--

---

**Description**

Load an rda or rds file and add it to the radiant data list (r\_data) if available

**Usage**

```
loadr(fn, objname = "")
```

**Arguments**

fn	File name and path as a string. Extension must be either rda or rds
objname	Name to use for the data frame. Defaults to the file name

**Value**

Data frame in r\_data or in the calling enviroment

---

loadrda_url	<i>Load an rda file from a url</i>
-------------	------------------------------------

---

**Description**

Load an rda file from a url

**Usage**

```
loadrda_url(rda_url)
```

**Arguments**

rda_url	URL for the csv file
---------	----------------------

**Value**

Data frame

make\_dt

*Make a pivot tabel in DT***Description**

Make a pivot tabel in DT

**Usage**

```
make_dt(pvt, format = "none", perc = FALSE, dec = 3, search = "",
        searchCols = NULL, order = NULL)
```

**Arguments**

pvt	Return value from <a href="#">pivotr</a>
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
search	Global search. Used to save and restore state
searchCols	Column search and filter. Used to save and restore state
order	Column sorting. Used to save and restore state

**Details**

See <http://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") %>% make_dt
pivotr("diamonds", cvars = c("cut","clarity")) %>% make_dt(format = "color_bar")
ret <- pivotr("diamonds", cvars = c("cut","clarity"), normalize = "total") %>%
  make_dt(format = "color_bar", perc = TRUE)
```

---

make_expl	<i>Make a tabel of summary statistics in DT</i>
-----------	---

---

## Description

Make a tabel of summary statistics in DT

## Usage

```
make_expl(expl, top = "fun", dec = 3, search = "", searchCols = NULL,  
          order = NULL)
```

## Arguments

expl	Return value from <a href="#">explore</a>
top	The variable (type) to display at the top of the table ("fun" for Function, "var" for Variable, and "byvar" for Group by)
dec	Number of decimals to show
search	Global search. Used to save and restore state
searchCols	Column search and filter. Used to save and restore state
order	Column sorting. Used to save and restore state

## Details

See <http://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") %>% make_expl  
tab <- explore("diamonds", "price", byvar = "cut", fun = c("length", "skew")) %>%  
  make_expl(top = "byvar")
```

---

make_funs	<i>Make a list of functions-as-formulas to pass to dplyr</i>
-----------	--

---

**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	<i>Generate a variable used to selected a training sample</i>
------------	---

---

**Description**

Generate a variable used to selected a training sample

**Usage**

```
make_train(n = 0.7, nr = 100)
```

**Arguments**

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

**Value**

0/1 variables for filtering

**Examples**

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

---

max_rm	<i>Max with na.rm = TRUE</i>
--------	------------------------------

---

**Description**

Max with na.rm = TRUE

**Usage**

```
max_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Maximum value

**Examples**

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

---

mean_rm	<i>Mean with na.rm = TRUE</i>
---------	-------------------------------

---

**Description**

Mean with na.rm = TRUE

**Usage**

```
mean_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Mean value

**Examples**

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

---

median_rm	<i>Median with na.rm = TRUE</i>
-----------	---------------------------------

---

**Description**

Median with na.rm = TRUE

**Usage**

```
median_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Median value

**Examples**

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

---

min_rm	<i>Min with na.rm = TRUE</i>
--------	------------------------------

---

**Description**

Min with na.rm = TRUE

**Usage**

```
min_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Minimum value

**Examples**

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

---

mode_rm	<i>Mode with na.rm = TRUE</i>
---------	-------------------------------

---

**Description**

Mode with na.rm = TRUE

**Usage**

```
mode_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Mode value

**Examples**

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

---

mutate_each	<i>Add transformed variables to a data frame (NSE)</i>
-------------	--

---

**Description**

Add tranformed variables to a data frame (NSE)

**Usage**

```
mutate_each(tbl, funs, ..., ext = "")
```

**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

**Details**

Wrapper for dplyr::mutate\_each that allows custom variable name extensions

**Examples**

```
mutate_each(mtcars, funs(log), mpg, cyl, ext = "_log")
```

---

normalize	<i>Normalize a variable <math>x</math> by a variable <math>y</math></i>
-----------	---

---

**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	<i>Number of missing values</i>
-----------	---------------------------------

---

**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	<i>5th percentile</i>
-----	-----------------------

---

**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	<i>10th percentile</i>
-----	------------------------

---

**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 = "", 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")
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 <http://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	<i>Don't try to plot strings</i>
----------------	----------------------------------

---

**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	<i>Plot method for the pivotr function</i>
-------------	--

---

**Description**

Plot method for the pivotr function

**Usage**

```
## S3 method for class 'pivotr'
plot(x, type = "dodge", perc = FALSE, flip = FALSE,
     shiny = FALSE, custom = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">pivotr</a>
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)
shiny	Did the function call originate inside a shiny app
custom	Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and <a href="http://docs.ggplot2.org/">http://docs.ggplot2.org/</a> for options.
...	further arguments passed to or from other methods

**Details**

See <http://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
```

---

<code>print.gtable</code>	<i>Print/draw method for grobs produced by gridExtra</i>
---------------------------	--

---

**Description**

Print/draw method for grobs produced by gridExtra

**Usage**

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

**Arguments**

<code>x</code>	a gtable object
<code>...</code>	further arguments passed to or from other methods

**Details**

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

**Value**

A plot

---

<code>publishers</code>	<i>Comic publishers</i>
-------------------------	-------------------------

---

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

---

<code>radiant.data</code>	<i><code>radiant.data</code></i>
---------------------------	----------------------------------

---

**Description**

`radiant.data`

Launch Radiant in the default browser

**Usage**

`radiant.data()`

**Details**

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

---

<code>radiant.data-deprecated</code>	<i>Deprecated function(s) in the <code>radiant.data</code> package</i>
--------------------------------------	--

---

**Description**

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

**Usage**

`dfprint(...)`

**Arguments**

... Parameters to be passed to the updated functions

**Details**

`dfprint` is now a synonym for `formatdf`  
`nrprint` is now a synonym for `formatnr`

---

render	<i>Method to render htmlwidgets</i>
--------	-------------------------------------

---

**Description**

Method to render htmlwidgets

**Usage**

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

**Arguments**

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

---

render.datatables	<i>Method to render DT tabels</i>
-------------------	-----------------------------------

---

**Description**

Method to render DT tabels

**Usage**

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

**Arguments**

object	DT table plot
...	Additional arguments

---

rounddf	<i>Round double in a data.frame to a specified number of decimal places</i>
---------	---

---

**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 = c("a","b"), y = c(1L, 2L), z = c(-0.0005, 3.1)) %>%  
  rounddf(dec = 3)
```

---

rownames_to_column	<i>Exporting rownames_to_column from tibble</i>
--------------------	---

---

**Description**

Exporting rownames\_to\_column from tibble

---

saver	<i>Save data.frame as an rda or rds file from Radiant</i>
-------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

Data frame in r\_data

---

sdp_rm	<i>Standard deviation for the population na.rm = TRUE</i>
--------	---

---

**Description**

Standard deviation for the population na.rm = TRUE

**Usage**

```
sdp_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Standard deviation for the population

**Examples**

```
sdp_rm(rnorm(100))
```

---

sd_rm	<i>Standard deviation with na.rm = TRUE</i>
-------	---

---

**Description**

Standard deviation with na.rm = TRUE

**Usage**

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

**Arguments**

x	Input variable
na.rm	Remove NAs (TRUE or FALSE)

**Value**

Standard deviation

**Examples**

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

---

serr	<i>Standard error</i>
------	-----------------------

---

**Description**

Standard error

**Usage**

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

**Arguments**

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

**Value**

Standard error

**Examples**

```
serr(rnorm(100))
```

---

set_attr	<i>Alias used to add an attribute</i>
----------	---------------------------------------

---

**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	<i>Show all rows with duplicated values (not just the first or last)</i>
-----------------	--

---

### 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	<i>Add stars '***' to a data.frame (from broom's 'tidy' function) based on p.values</i>
-----------	---

---

### 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	<i>Exporting the skew function from the psych package</i>
------	---

---

**Description**

Exporting the skew function from the psych package

---

square	<i>Calculate square of a variable</i>
--------	---------------------------------------

---

**Description**

Calculate square of a variable

**Usage**

```
square(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

$x^2$

---

`ssh`*Hide warnings and messages and return invisible*

---

**Description**

Hide warnings and messages and return invisible

**Usage**

```
ssh(...)
```

**Arguments**

...                      Inputs to keep quiet

**Details**

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

**Examples**

```
ssh( library(dplyr) )
```

---

`sshhr`*Hide warnings and messages and return result*

---

**Description**

Hide warnings and messages and return result

**Usage**

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

**Arguments**

...                      Inputs to keep quiet

**Details**

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

**Examples**

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

---

standardize	<i>Standardize</i>
-------------	--------------------

---

**Description**

Standardize

**Usage**

```
standardize(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

If x is a numeric variable return  $\text{center}(x) / \text{mean}(x)$

---

store	<i>Method to store variables in a dataset in Radiant</i>
-------	--

---

**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

---

summary.explore	<i>Summary method for the explore function</i>
-----------------	--

---

**Description**

Summary method for the explore function

**Usage**

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

**Arguments**

object	Return value from <a href="#">explore</a>
top	The variable (type) to display at the top of the table
dec	Number of decimals to show
...	further arguments passed to or from other methods

**Details**

See <http://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	<i>Summary method for pivotr</i>
----------------	----------------------------------

---

**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 <a href="#">pivotr</a>
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 <http://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", 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)
```

**Arguments**

x	Input variable
---	----------------

**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](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	<i>Exporting tidy from broom</i>
------	----------------------------------

---

**Description**

Exporting tidy from broom

---

titanic	<i>Survival data for the Titanic</i>
---------	--------------------------------------

---

**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")

---

varp_rm	<i>Variance for the population na.rm = TRUE</i>
---------	---

---

**Description**

Variance for the population na.rm = TRUE

**Usage**

```
varp_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Variance for the population

**Examples**

```
varp_rm(rnorm(100))
```

---

var_rm	Variance with na.rm = TRUE
--------	----------------------------

---

**Description**

Variance with na.rm = TRUE

**Usage**

```
var_rm(x)
```

**Arguments**

x	Input variable
---	----------------

**Value**

Variance

**Examples**

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

---

viewdata	View data
----------	-----------

---

**Description**

View data

**Usage**

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

**Arguments**

dataset	Name of the dataframe to change
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://docs.ggplot2.org/current/>


---

## Description

Visualize data using ggplot2 <http://docs.ggplot2.org/current/>

## Usage

```
visualize(dataset, xvar, yvar = "", comby = FALSE, combx = FALSE,
  type = "dist", facet_row = ".", facet_col = ".", color = "none",
  fill = "none", bins = 10, smooth = 1, fun = "mean", check = "",
  axes = "", alpha = 0.5, 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 <code>r_data</code> 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'), 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
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)
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 option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and <a href="http://docs.ggplot2.org/">http://docs.ggplot2.org/</a> for options.

### Details

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

### Value

Generated plots

### Examples

```
visualize("diamonds", "carat", "price", type = "scatter", check = "loess")
visualize("diamonds", "price:x", type = "dist")
visualize("diamonds", "carat:x", yvar = "price", type = "scatter")
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", custom = TRUE) +
  ggtitle("A scatterplot") + xlab("price in $")
visualize(dataset = "diamonds", xvar = "price:carat", custom = TRUE) %>%
  {.[[1]] + ggtitle("A histogram") + xlab("price in $")}
diamonds %>% visualize(c("price", "carat", "depth"), type = "density")
visualize(dataset = "diamonds", xvar = "cut", yvar = "price", type = "bar",
  facet_row = "cut", fill = "cut", custom = FALSE)
visualize(dataset = "diamonds", xvar = "cut", yvar = "price", type = "line",
  facet_row = "cut", color = "cut", custom = FALSE)
```

---

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	<i>Returns the index of the (parallel) maxima of the input values</i>
------------	---

---

**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	<i>Returns the index of the (parallel) minima of the input values</i>
------------	---

---

**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 a quintile (or decile) index*

---

**Description**

Create a quintile (or decile) index

**Usage**

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

**Arguments**

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

**Details**

Same as stata

**Examples**

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



# Index

## \*Topic **datasets**

- avengers, [14](#)
- diamonds, [20](#)
- publishers, [46](#)
- superheroes, [58](#)
- titanic, [59](#)

- add\_class, [4](#)
- as\_character, [5](#)
- as\_data\_frame, [5](#)
- as\_distance, [5](#)
- as\_dmy, [6](#)
- as\_dmy\_hm, [6](#)
- as\_dmy\_hms, [7](#)
- as\_duration, [7](#)
- as\_factor, [8](#)
- as\_hm, [8](#)
- as\_hms, [9](#)
- as\_integer, [9](#)
- as\_mdy, [10](#)
- as\_mdy\_hm, [11](#)
- as\_mdy\_hms, [11](#)
- as\_numeric, [12](#)
- as\_tibble, [12](#)
- as\_ymd, [12](#)
- as\_ymd\_hm, [13](#)
- as\_ymd\_hms, [13](#)
- avengers, [14](#)

- center, [14](#)
- changedata, [15](#)
- ci\_label, [15](#)
- ci\_perc, [16](#)
- combinedata, [16](#)
- copy\_all, [17](#)
- copy\_from, [18](#)
- cv, [19](#)

- data\_frame, [19](#)
- describe, [19](#)
- dfprint (radiant.data-deprecated), [47](#)
- diamonds, [20](#)
- does\_vary, [20](#)

- explore, [21](#), [23](#), [35](#), [56](#)

- factorizer, [22](#)
- filterdata, [22](#)
- find\_dropbox, [23](#)
- flip, [23](#)
- formatdf, [24](#), [47](#)
- formatnr, [24](#), [47](#)

- getclass, [25](#)
- getdata, [26](#)
- getsummary, [26](#)
- glance, [27](#)

- indexr, [27](#)
- install\_webshot, [27](#)
- inverse, [28](#)
- is\_empty, [28](#)
- is\_not, [29](#)
- is\_string, [29](#)
- iterms, [30](#)

- kurtosi, [30](#)

- level\_list, [31](#)
- ln, [31](#)
- loadcsv, [32](#)
- loadcsv\_url, [32](#)
- loadr, [33](#)
- loadrda\_url, [33](#)

- make\_dt, [34](#)
- make\_expl, [23](#), [35](#)
- make\_funs, [36](#)
- make\_train, [36](#)
- max\_rm, [37](#)
- mean\_rm, [37](#)
- median\_rm, [38](#)
- min\_rm, [38](#)
- mode\_rm, [39](#)
- mutate\_each, [39](#)

- n\_missing, [40](#)
- normalize, [40](#)
- nrprint (radiant.data-deprecated), [47](#)

- p05, [41](#)

p10, [41](#)  
p25, [42](#)  
p75, [42](#)  
p90, [43](#)  
p95, [43](#)  
pivotr, [34](#), [35](#), [44](#), [45](#), [46](#), [57](#)  
plot.character, [45](#)  
plot.pivotr, [45](#)  
print.gtable, [46](#)  
publishers, [46](#)  
  
radiant.data, [47](#)  
radiant.data-deprecated, [47](#)  
radiant.data-deprecated-package  
    (radiant.data-deprecated), [47](#)  
radiant.data-package (radiant.data), [47](#)  
render, [48](#)  
render.datatables, [48](#)  
rounddf, [48](#)  
rownames\_to\_column, [49](#)  
  
saver, [49](#)  
sd\_rm, [50](#)  
sdp\_rm, [50](#)  
serr, [51](#)  
set\_attr, [51](#)  
show\_duplicated, [52](#)  
sig\_stars, [52](#)  
skew, [53](#)  
square, [53](#)  
sshh, [54](#)  
sshhr, [54](#)  
standardize, [55](#)  
store, [55](#)  
sum\_rm, [57](#)  
summary.explore, [21](#), [56](#)  
summary.pivotr, [34](#), [35](#), [46](#), [56](#)  
superheroes, [58](#)  
  
table2data, [58](#)  
tibble, [58](#)  
tidy, [59](#)  
titanic, [59](#)  
  
var\_rm, [60](#)  
varp\_rm, [59](#)  
viewdata, [60](#)  
visualize, [61](#)  
  
weighted.sd, [62](#)  
which.pmax, [63](#)  
which.pmin, [63](#)  
  
xtile, [64](#)