

# Package ‘radiant’

August 16, 2015

**Title** Business Analytics using R and Shiny

**Version** 0.3.0

**Date** 2015-8-16

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

**Depends** R (>= 3.2.0),  
magrittr (>= 1.5),  
ggplot2 (>= 1.0.0),  
tidyr (>= 0.2.0),  
dplyr (>= 0.4.2)

**Imports** DiagrammeR(>= 0.7),  
car (>= 2.0.22),  
MASS (>= 7.3),  
gridExtra (>= 2.0.0),  
AlgDesign (>= 1.1.7.3),  
psych (>= 1.4.8.11),  
GPArotation (>= 2014.11.1),  
wordcloud (>= 2.5),  
markdown (>= 0.7.4),  
rmarkdown (>= 0.4.2),  
knitr (>= 1.8),  
ggdendro (>= 0.1.15),  
broom (>= 0.3.6),  
pryr (>= 0.1),  
shiny (>= 0.12.2),  
shinyAce (>= 0.2.1),  
lubridate (>= 1.3.3),  
DT (>= 0.1.32),  
MathJaxR (>= 0.11),  
readr (>= 0.1.1),  
data.tree(>= 0.1.9),  
yaml(>= 2.1.13),  
scales(>= 0.2.5)

**Suggests** ggvis (>= 0.4),  
devtools (>= 1.7.0),  
testthat (>= 0.9.1),  
covr (>= 1.2.0)

**URL** <https://github.com/vnijs/radiant>, <http://vnijs.github.io/radiant/>

**BugReports** <https://github.com/vnijs/radiant/issues>

**License** AGPL-3 | file LICENSE

**LazyData** true

## R topics documented:

avengers . . . . .	4
changedata . . . . .	5
city . . . . .	6
clean_loadings . . . . .	6
combinedata . . . . .	7
compare_means . . . . .	8
compare_props . . . . .	9
computer . . . . .	10
conjoint . . . . .	10
conjoint_profiles . . . . .	11
copy_all . . . . .	12
copy_from . . . . .	12
correlation . . . . .	13
cross_tabs . . . . .	14
cv . . . . .	15
diamonds . . . . .	15
dtree . . . . .	16
explore . . . . .	16
factorizer . . . . .	17
ff_design . . . . .	18
filterdata . . . . .	18
flip . . . . .	19
full_factor . . . . .	19
getclass . . . . .	20
getdata . . . . .	21
getsummary . . . . .	22
glm_reg . . . . .	22
hier_clus . . . . .	23
is_empty . . . . .	24
is_string . . . . .	25
iterms . . . . .	25
kmeans_clus . . . . .	26
kurtosi . . . . .	27
launcher . . . . .	27
lin_launcher . . . . .	28
loadcsv . . . . .	28
mac_launcher . . . . .	29
make_dt . . . . .	30
make_expl . . . . .	30
make_funs . . . . .	31
max_rm . . . . .	31
mds . . . . .	32
mean_rm . . . . .	33
median_rm . . . . .	33
min_rm . . . . .	34

mp3	34
newspaper	35
nmissing	35
p05	36
p25	36
p75	37
p95	37
pivotr	38
plot.compare_means	38
plot.compare_props	39
plot.conjoint	40
plot.correlation_	40
plot.cross_tabs	41
plot.dtree	42
plot.full_factor	43
plot.glm_predict	43
plot.glm_reg	44
plot.hier_clus	45
plot.kmeans_clus	46
plot.mds	47
plot.pivotr	48
plot.pmap	48
plot.pre_factor	49
plot.regression	50
plot.reg_predict	51
plot.repeater	52
plot.simulater	53
plot.single_mean	53
plot.single_prop	54
pmap	55
predict.glm_reg	56
predict.regression	57
pre_factor	58
print.gtable	58
publishers	59
radiant	59
regression	60
repeater	61
rndnames	61
sample_size	62
sampling	63
save_factors	63
save_membership	64
sd_rm	65
serr	65
set_class	66
shopping	66
sig_stars	66
simulater	67
single_mean	68
single_prop	69
skew	70

sshh . . . . .	70
sshhr . . . . .	70
state_init . . . . .	71
state_multiple . . . . .	72
state_single . . . . .	73
store_glm . . . . .	74
store_reg . . . . .	74
summary.compare_means . . . . .	75
summary.compare_props . . . . .	76
summary.conjoint . . . . .	76
summary.conjoint_profiles . . . . .	77
summary.correlation_ . . . . .	78
summary.cross_tabs . . . . .	78
summary.dtree . . . . .	79
summary.explore . . . . .	80
summary.full_factor . . . . .	80
summary.glm_reg . . . . .	81
summary.hier_clus . . . . .	82
summary.kmeans_clus . . . . .	83
summary.mds . . . . .	83
summary.pivotr . . . . .	84
summary.pmap . . . . .	85
summary.pre_factor . . . . .	86
summary.regression . . . . .	86
summary.repeater . . . . .	87
summary.sample_size . . . . .	88
summary.sampling . . . . .	88
summary.simulater . . . . .	89
summary.single_mean . . . . .	90
summary.single_prop . . . . .	90
sum_rm . . . . .	91
superheroes . . . . .	91
test_specs . . . . .	92
the_table . . . . .	92
titanic . . . . .	93
titanic_pred . . . . .	93
toothpaste . . . . .	94
var_check . . . . .	94
viewdata . . . . .	95
visualize . . . . .	95
win_launcher . . . . .	97

<b>Index</b>	<b>98</b>
--------------	-----------

---

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")`

---

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

**Examples**

```
r_data <- list()
r_data$dat <- data.frame(a = 1:20)
changedata("dat", 20:1, "b")
head(r_data$dat)
rm(r_data, envir = .GlobalEnv)
```

---

city	<i>City distances</i>
------	-----------------------

---

**Description**

City distances

**Usage**

```
data(city)
```

**Format**

A data frame with 45 rows and 3 variables

**Details**

Distance in miles between nine cities in the USA. The dataset is used to illustrate multi-dimensional scaling (MDS). Description provided in `attr(city,"description")`

---

clean_loadings	<i>Sort and clean loadings</i>
----------------	--------------------------------

---

**Description**

Sort and clean loadings

**Usage**

```
clean_loadings(floadings, cutoff = 0, fsort = FALSE, dec = 8)
```

**Arguments**

floadings	Data frame with loadings
cutoff	Show only loadings with (absolute) values above cutoff (default = 0)
fsort	Sort factor loadings
dec	Number of decimals to show

**Details**

See [http://vnijs.github.io/radiant/marketing/full\\_factor.html](http://vnijs.github.io/radiant/marketing/full_factor.html) for an example in Radiant

**Examples**

```
result <- full_factor("diamonds",c("price","carat","table","x","y"))
clean_loadings(result$floadings, TRUE, .5, 2)
```

combinedata

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

Combine datasets using dplyr's bind and join functions

**Usage**

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

**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'
type	The main bind and join types from the dplyr package are provided. <b>inner_join</b> 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. <b>left_join</b> 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. <b>right_join</b> is equivalent to a left join for datasets y and x. <b>full_join</b> combines two datasets, keeping rows and columns that appear in either. <b>semi_join</b> 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. <b>anti_join</b> returns all rows from x without matching values in y, keeping only columns from x. <b>bind_rows</b> and <b>bind_cols</b> are also included, as are <b>intersect</b> , <b>union</b> , and <b>setdiff</b> . See <a href="http://vnijs.github.io/radiant/base/combine.html">http://vnijs.github.io/radiant/base/combine.html</a> for further details
name	Name for the combined dataset

**Details**

See <http://vnijs.github.io/radiant/base/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**

```
combinedata("titanic", "titanic_pred", c("pclass", "sex", "age")) %>% head
titanic %>% combinedata("titanic_pred", c("pclass", "sex", "age")) %>% head
titanic %>% combinedata(titanic_pred, c("pclass", "sex", "age")) %>% head
avengers %>% combinedata(superheroes, type = "bind_cols")
combinedata("avengers", "superheroes", type = "bind_cols")
avengers %>% combinedata(superheroes, type = "bind_rows")
```

---

compare\_means

---

*Compare means for two or more variables*


---

## Description

Compare means for two or more variables

## Usage

```
compare_means(dataset, var1, var2, samples = "independent",
  alternative = "two.sided", conf_lev = 0.95, adjust = "none",
  test = "t", 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
var1	A numeric variable or factor selected for comparison
var2	One or more numeric variables for comparison. If var1 is a factor only one variable can be selected and the mean of this variable is compared across (factor) levels of var1
samples	Are samples indepent ("independent") or not ("paired")
alternative	The alternative hypothesis ("two.sided", "greater" or "less")
conf_lev	Span of the confidence interval
adjust	Adjustment for multiple comparisons ("none" or "bonf" for Bonferroni)
test	T-test ("t") or Wilcox ("wilcox")
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")

## Details

See [http://vnijs.github.io/radiant/quant/compare\\_means.html](http://vnijs.github.io/radiant/quant/compare_means.html) for an example in Radiant

## Value

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

## See Also

[summary.compare\\_means](#) to summarize results

[plot.compare\\_means](#) to plot results

## Examples

```
result <- compare_means("diamonds", "cut", "price")
result <- diamonds %>% compare_means("cut", "price")
```



compare\_props

*Compare proportions across groups***Description**

Compare proportions across groups

**Usage**

```
compare_props(dataset, var1, var2, levs = "", alternative = "two.sided",
  conf_lev = 0.95, adjust = "none", 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
var1	A grouping variable to split the data for comparisons
var2	The variable to calculate proportions for
levs	The factor level selected for the proportion comparison
alternative	The alternative hypothesis ("two.sided", "greater" or "less")
conf_lev	Span of the confidence interval
adjust	Adjustment for multiple comparisons ("none" or "bonf" for Bonferroni)
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")

**Details**

See [http://vnijs.github.io/radiant/quant/compare\\_props.html](http://vnijs.github.io/radiant/quant/compare_props.html) for an example in Radiant

**Value**

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

**See Also**

[summary.compare\\_props](#) to summarize results

[plot.compare\\_props](#) to plot results

**Examples**

```
result <- compare_props("titanic", "pclass", "survived")
result <- titanic %>% compare_props("pclass", "survived")
```

---

computer	<i>Perceptions of computer (re)sellers</i>
----------	--------------------------------------------

---

**Description**

Perceptions of computer (re)sellers

**Usage**

```
data(computer)
```

**Format**

A data frame with 5 rows and 8 variables

**Details**

Perceptions of computer (re)sellers. The dataset is used to illustrate perceptual maps. Description provided in `attr(computer,"description")`

---

conjoint	<i>Conjoint analysis</i>
----------	--------------------------

---

**Description**

Conjoint analysis

**Usage**

```
conjoint(dataset, dep_var, indep_var, reverse = FALSE, 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
dep_var	The dependent variable (e.g., profile ratings)
indep_var	Independent variables in the regression
reverse	Reverse the values of the dependent variable ('dep_var')
data_filter	Expression entered in, e.g., <code>Data &gt; View</code> to filter the dataset in Radiant. The expression should be a string (e.g., <code>"price &gt; 10000"</code> )

**Details**

See <http://vnijs.github.io/radiant/marketing/conjoint.html> for an example in Radiant

**Value**

A list with all variables defined in the function as an object of class `conjoint`

**See Also**

[summary.conjoint](#) to summarize results

[plot.conjoint](#) to plot results

**Examples**

```
result <- conjoint("mp3", dep_var = "Rating", indep_var = "Memory:Shape")
result <- mp3 %>% conjoint(dep_var = "Rating", indep_var = "Memory:Shape")
```

---

conjoint_profiles	<i>Create fractional factorial design for conjoint analysis</i>
-------------------	-----------------------------------------------------------------

---

**Description**

Create fractional factorial design for conjoint analysis

**Usage**

```
conjoint_profiles(dataset)
```

**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
---------	------------------------------------------------------------------------------------------------------------------------------------

**Details**

See [http://vnijs.github.io/radiant/marketing/conjoint\\_profiles.html](http://vnijs.github.io/radiant/marketing/conjoint_profiles.html) for an example in Radiant

**Value**

A list with all variables defined in the function as an object of class `conjoint_profiles`

**See Also**

[summary.conjoint\\_profiles](#) to summarize results

**Examples**

```
cp <- readLines(system.file("examples/profiles-movie.txt", package='radiant'))
result <- conjoint_profiles("cp")
rm(cp, envir = .GlobalEnv)
result <- readLines(system.file("examples/profiles-movie.txt", package='radiant')) %>%
  conjoint_profiles
```

---

copy_all	<i>Source all package functions</i>
----------	-------------------------------------

---

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

**Examples**

```
copy_all(radiant)
```

---

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, state_init)
```

---

correlation	<i>Calculate correlations for two or more variables</i>
-------------	---------------------------------------------------------

---

### Description

Calculate correlations for two or more variables

### Usage

```
correlation(dataset, vars, type = "pearson", 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
vars	Variables to include in the analysis
type	Type of correlations to calculate. Options are "pearson", "spearman", and "kendall". "pearson" is the default
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")

### Details

See <http://vnijs.github.io/radiant/quant/correlation.html> for an example in Radiant

### Value

A list with all variables defined in the function as an object of class `compare_means`

### See Also

[summary.correlation\\_](#) to summarize results

[plot.correlation\\_](#) to plot results

### Examples

```
result <- correlation("diamonds", c("price","carat","clarity"))
result <- correlation("diamonds", "price:table")
result <- diamonds %>% correlation("price:table")
```

---

`cross_tabs`*Evaluate associations between categorical variables*

---

## Description

Evaluate associations between categorical variables

## Usage

```
cross_tabs(dataset, var1, var2, data_filter = "")
```

## Arguments

<code>dataset</code>	Dataset name (string). This can be a dataframe in the global environment or an element in an <code>r_data</code> list from Radiant
<code>var1</code>	A categorical variable
<code>var2</code>	Another categorical variable
<code>data_filter</code>	Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")

## Details

See [http://vnijs.github.io/radiant/quant/cross\\_tabs.html](http://vnijs.github.io/radiant/quant/cross_tabs.html) for an example in Radiant

## Value

A list of all variables used in `cross_tabs` as an object of class `cross_tabs`

## See Also

[`summary.cross\_tabs`](#) to summarize results

[`plot.cross\_tabs`](#) to plot results

## Examples

```
result <- cross_tabs("newspaper", "Income", "Newspaper")
result <- newspaper %>% cross_tabs("Income", "Newspaper")
```

---

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))
```

---

diamonds	<i>Diamond prices</i>
----------	-----------------------

---

**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")`

---

dtree	<i>Create a decision tree</i>
-------	-------------------------------

---

**Description**

Create a decision tree

**Usage**

```
dtree(y1)
```

**Arguments**

y1                      A yaml string or a list (e.g., from `yaml::yaml.load_file()`)

**Details**

See <http://vnijs.github.io/radiant/base/dtree.html> for an example in Radiant

**Value**

A list with the initial tree and the calculated tree

**See Also**

`summary.dtree` to summarize results

`plot.dtree` to plot results

---

explore	<i>Explore data</i>
---------	---------------------

---

**Description**

Explore data

**Usage**

```
explore(dataset, vars = "", byvar = "", fun = "mean_rm",
  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
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://vnijs.github.io/radiant/base/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", "price", byvar = "cut", fun = c("length", "skew"))
summary(result)
diamonds %>% explore("price", byvar = "cut", fun = c("length", "skew"))
```

---

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

---

## Description

Convert character to factors as needed

## Usage

```
factorizer(dat, safx = 10)
```

## Arguments

<code>dat</code>	Data.frame
<code>safx</code>	Values to levels ratio

## Value

Data.frame with factors

---

ff_design	<i>Function to generate a fractional factorial design</i>
-----------	-----------------------------------------------------------

---

### Description

Function to generate a fractional factorial design

### Usage

```
ff_design(attr, trial = 0, rseed = 172110)
```

### Arguments

attr	Attributes used to generate profiles
trial	Number of trials that have already been run
rseed	Random seed to use

### Details

See [http://vnijs.github.io/radiant/marketing/conjoint\\_profiles.html](http://vnijs.github.io/radiant/marketing/conjoint_profiles.html) for an example in Radiant

### See Also

[conjoint\\_profiles](#) to calculate results  
[summary.conjoint\\_profiles](#) to summarize results

---

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

---

 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)
```

### 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)

### Details

See <http://vnijs.github.io/radiant/base/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")
```

---

 full\_factor

---

*Factor analysis (PCA)*


---

### Description

Factor analysis (PCA)

### Usage

```
full_factor(dataset, vars, method = "PCA", nr_fact = 2,
  rotation = "varimax", 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
vars	Variables to include in the analysis
method	Factor extraction method to use
nr_fact	Number of factors to extract
rotation	Apply varimax rotation or no rotation ("varimax" or "none")
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")

**Details**

See [http://vnijs.github.io/radiant/marketing/full\\_factor.html](http://vnijs.github.io/radiant/marketing/full_factor.html) for an example in Radiant

**Value**

A list with all variables defined in the function as an object of class `full_factor`

**See Also**

`summary.full_factor` to summarize results

`plot.full_factor` to plot results

**Examples**

```
result <- full_factor("diamonds",c("price","carat","table","x","y"))
result <- full_factor("diamonds",c("price","carat","table","x","y"), method = "maxlik")
result <- diamonds %>% full_factor(c("price","carat","table","x","y"), method = "maxlik")
```

---

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

**Examples**

```
r_data <- list()
r_data$dat <- mtcars
getdata("dat", "mpg:vs", filt = "mpg > 20", rows = 1:5)
rm(r_data, envir = .GlobalEnv)
```

---

getsummary	Create data.frame summary
------------	---------------------------

---

**Description**

Create data.frame summary

**Usage**

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

**Arguments**

dat	Data.frame
dc	Class for each variable

**Details**

Used by Transform

---

glm_reg	Generalized linear models (GLM)
---------	---------------------------------

---

**Description**

Generalized linear models (GLM)

**Usage**

```
glm_reg(dataset, dep_var, indep_var, lev = "", link = "logit",
  int_var = "", check = "", 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
dep_var	The dependent variable in the logit (probit) model
indep_var	Independent variables in the model
lev	The level in the dependent variable defined as <code>_success_</code>
link	Link function for <code>_glm_</code> ('logit' or 'probit'). 'logit' is the default
int_var	Interaction term to include in the model (not implement)
check	Optional output or estimation parameters. "vif" to show the multicollinearity diagnostics. "confint" to show coefficient confidence interval estimates. "odds" to show odds ratios and confidence interval estimates. "standardize" to output standardized coefficient estimates. "stepwise" to apply step-wise selection of variables
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")

**Details**

See [http://vnijs.github.io/radiant/quant/glm\\_reg.html](http://vnijs.github.io/radiant/quant/glm_reg.html) for an example in Radiant

**Value**

A list with all variables defined in glm\_reg as an object of class glm\_reg

**See Also**

`summary.glm_reg` to summarize the results  
`plot.glm_reg` to plot the results  
`predict.glm_reg` to generate predictions  
`plot.glm_predict` to plot prediction output

**Examples**

```
result <- glm_reg("titanic", "survived", c("pclass","sex"), lev = "Yes")
result <- glm_reg("titanic", "survived", c("pclass","sex"))
```

---

hier_clus	<i>Hierarchical cluster analysis</i>
-----------	--------------------------------------

---

**Description**

Hierarchical cluster analysis

**Usage**

```
hier_clus(dataset, vars, distance = "sq.euclidian", method = "ward.D",
  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
vars	Vector of variables to include in the analysis
distance	Distance
method	Method
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")

**Details**

See [http://vnijs.github.io/radiant/marketing/hier\\_clus.html](http://vnijs.github.io/radiant/marketing/hier_clus.html) for an example in Radiant

**Value**

A list of all variables used in hier\_clus as an object of class hier\_clus

**See Also**

[summary.hier\\_clus](#) to summarize results

[plot.hier\\_clus](#) to plot results

**Examples**

```
result <- hier_clus("shopping", vars = c("v1:v6"))
```

---

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

---

**Description**

Is a character variable defined

**Usage**

```
is_empty(x, empty = "")
```

**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_string	<i>Is input a string?</i>
-----------	---------------------------

---

**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	<i>Create a vector of interaction terms</i>
--------	---------------------------------------------

---

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

kmeans\_clus

*K-means cluster analysis***Description**

K-means cluster analysis

**Usage**

```
kmeans_clus(dataset, vars, hc_init = TRUE, distance = "sq.euclidian",
  method = "ward.D", seed = 1234, nr_clus = 2, 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
vars	Vector of variables to include in the analysis
hc_init	Use centers from <code>hier_clus</code> as the starting point
distance	Distance for <code>hier_clus</code>
method	Method for <code>hier_clus</code>
seed	Random seed to use for <code>kmeans</code> if <code>hc_init</code> is FALSE
nr_clus	Number of clusters to extract
data_filter	Expression entered in, e.g., <code>Data &gt; View</code> to filter the dataset in Radiant. The expression should be a string (e.g., <code>"price &gt; 10000"</code> )

**Details**

See [http://vnijs.github.io/radiant/marketing/kmeans\\_clus.html](http://vnijs.github.io/radiant/marketing/kmeans_clus.html) for an example in Radiant

**Value**

A list of all variables used in `kmeans_clus` as an object of class `kmeans_clus`

**See Also**

[summary.kmeans\\_clus](#) to summarize results  
[plot.kmeans\\_clus](#) to plot results  
[save\\_membership](#) to add cluster membership to the selected dataset

**Examples**

```
result <- kmeans_clus("shopping", c("v1:v6"))
```

---

kurtosi	<i>Exporting the kurtosi function from the psych package</i>
---------	--------------------------------------------------------------

---

### Description

Exporting the kurtosi function from the psych package

---

launcher	<i>Create a launcher on the desktop for Windows (.bat), Mac (.command), or Linux (.sh)</i>
----------	--------------------------------------------------------------------------------------------

---

### Description

Create a launcher on the desktop for Windows (.bat), Mac (.command), or Linux (.sh)

### Usage

```
launcher(app = c("analytics", "marketing", "quant", "base"))
```

### Arguments

app	App to run when the desktop icon is double-clicked ("analytics", "marketing", "quant", or "base"). Default is "analytics"
-----	---------------------------------------------------------------------------------------------------------------------------

### Details

On Windows/Mac/Linux a file named radiant.bat/radiant.command/radiant.sh will be put on the desktop. Double-click the file to launch the specified Radiant app

### See Also

[win\\_launcher](#) to create a shortcut on Windows

[mac\\_launcher](#) to create a shortcut on Mac

[lin\\_launcher](#) to create a shortcut on Linux

---

lin_launcher	<i>Create a launcher and updater for Linux (.sh)</i>
--------------	------------------------------------------------------

---

### Description

Create a launcher and updater for Linux (.sh)

### Usage

```
lin_launcher(app = c("analytics", "marketing", "quant", "base"))
```

### Arguments

app	App to run when the desktop icon is double-clicked ("analytics", "marketing", "quant", or "base"). Default is "analytics"
-----	---------------------------------------------------------------------------------------------------------------------------

### Details

On Linux a file named 'radiant.sh' and one named 'update\_radiant.sh' will be put on the desktop. Double-click the file to launch the specified Radiant app or update Radiant to the latest version

### Examples

```
if (interactive()) {
  if (Sys.info()["sysname"] == "Linux") {
    lin_launcher()
    fn <- paste0("/home/", Sys.getenv("USER"), "/Desktop/radiant.sh")
    if (!file.exists(fn))
      stop("Linux launcher not created")
    else
      unlink(fn)
  }
}
```

---

loadcsv	<i>Load a csv files with read.csv and read_csv</i>
---------	----------------------------------------------------

---

### Description

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

### Usage

```
loadcsv(fn, header = TRUE, sep = ",", saf = TRUE, safx = 10)
```

**Arguments**

fn	File name string
header	Header in file (TRUE, FALSE)
sep	Use , or ; or \t
saf	Convert character variables to factors if (1) there are less than 100 distinct values (2) there are X (see safx) more values than levels
safx	Values to levels ratio

**Value**

Data.frame with (some) variables converted to factors

---

mac_launcher	<i>Create a launcher and updater for Mac (.command)</i>
--------------	---------------------------------------------------------

---

**Description**

Create a launcher and updater for Mac (.command)

**Usage**

```
mac_launcher(app = c("analytics", "marketing", "quant", "base"))
```

**Arguments**

app	App to run when the desktop icon is double-clicked ("analytics", "marketing", "quant", or "base"). Default is "analytics"
-----	---------------------------------------------------------------------------------------------------------------------------

**Details**

On Mac a file named 'radiant.command' and one named 'update\_radiant.command' will be put on the desktop. Double-click the file to launch the specified Radiant app or update Radiant to the latest version

**Examples**

```
if (interactive()) {
  if (Sys.info()["sysname"] == "Darwin") {
    mac_launcher()
    fn <- paste0("/Users/", Sys.getenv("USER"), "/Desktop/radiant.command")
    if (!file.exists(fn))
      stop("Mac launcher not created")
    else
      unlink(fn)
  }
}
```

---

make_dt	<i>Make a pivot tabel in DT</i>
---------	---------------------------------

---

### Description

Make a pivot tabel in DT

### Usage

```
make_dt(pvt, format = "none", perc = FALSE)
```

### 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)

### Details

See <http://vnijs.github.io/radiant/base/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)
```

### 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

**Details**

See <http://vnijs.github.io/radiant/base/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"))
```

---

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))
```

---

mds	<i>(Dis)similarity based brand maps (MDS)</i>
-----	-----------------------------------------------

---

**Description**

(Dis)similarity based brand maps (MDS)

**Usage**

```
mds(dataset, id1, id2, dis, method = "metric", nr_dim = 2,
      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
id1	A character variable or factor with unique entries
id2	A character variable or factor with unique entries
dis	A numeric measure of brand dissimilarity
method	Apply metric or non-metric MDS
nr_dim	Number of dimensions
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")

**Details**

See <http://vnijs.github.io/radiant/marketing/mds.html> for an example in Radiant

**Value**

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

**See Also**

[summary.mds](#) to summarize results

[plot.mds](#) to plot results

**Examples**

```
result <- mds("city", "from", "to", "distance")
summary(result)
result <- mds("diamonds", "clarity", "cut", "price")
summary(result)
```



---

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))
```

---

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

---

**Description**

Min with na.rm = TRUE

**Usage**

```
min_rm(x)
```

**Arguments**

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

**Value**

Minimum value

**Examples**

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

---

<code>mp3</code>	<i>Conjoint data for MP3 players</i>
------------------	--------------------------------------

---

**Description**

Conjoint data for MP3 players

**Usage**

```
data(mp3)
```

**Format**

A data frame with 18 rows and 6 variables

**Details**

Conjoint data for MP3 players. Description provided in `attr(mp3,"description")`

---

newspaper	<i>Newspaper readership</i>
-----------	-----------------------------

---

**Description**

Newspaper readership

**Usage**

```
data(newspaper)
```

**Format**

A data frame with 580 rows and 2 variables

**Details**

Newspaper readership data for 580 consumers. Description provided in `attr(newspaper,"description")`

---

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

---

**Description**

Number of missing values

**Usage**

```
nmissing(x)
```

**Arguments**

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

**Value**

number of missing values

**Examples**

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

---

p05	5th percentile
-----	----------------

---

**Description**

5th percentile

**Usage**

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

**Arguments**

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

**Value**

5th percentile

**Examples**

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

---

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))
```

---

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",
        normalize = "None", 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"
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://vnijs.github.io/radiant/base/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
```

---

plot.compare_means	Plot method for the compare_means function
--------------------	--------------------------------------------

---

**Description**

Plot method for the compare\_means function

**Usage**

```
## S3 method for class 'compare_means'
plot(x, plots = "bar", shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">compare_means</a>
plots	One or more plots ("bar", "box", or "density")
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/compare\\_means.html](http://vnijs.github.io/radiant/quant/compare_means.html) for an example in Radiant

**See Also**

[compare\\_means](#) to calculate results  
[summary.compare\\_means](#) to summarize results

**Examples**

```
result <- compare_means("diamonds", "cut", "price")
plot(result, plots = c("bar", "density"))
```

---

plot.compare_props	<i>Plot method for the compare_props function</i>
--------------------	---------------------------------------------------

---

**Description**

Plot method for the compare\_props function

**Usage**

```
## S3 method for class 'compare_props'
plot(x, plots = "props", shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">compare_props</a>
plots	One or more plots of proportions or counts ("props" or "counts")
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/compare\\_props.html](http://vnijs.github.io/radiant/quant/compare_props.html) for an example in Radiant

**See Also**

[compare\\_props](#) to calculate results  
[summary.compare\\_props](#) to summarize results

**Examples**

```
result <- compare_props("titanic", "pclass", "survived")
plot(result, plots = c("props", "counts"))
```

---

plot.conjoint	<i>Plot method for the conjoint function</i>
---------------	----------------------------------------------

---

**Description**

Plot method for the conjoint function

**Usage**

```
## S3 method for class 'conjoint'
plot(x, plots = "pw", scale_plot = FALSE,
     shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">conjoint</a>
plots	Show either the part-worth ("pw") or importance-weights ("iw") plot
scale_plot	Scale the axes of the part-worth plots to the same range
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/marketing/conjoint.html> for an example in Radiant

**See Also**

[conjoint](#) to generate results  
[summary.conjoint](#) to summarize results

**Examples**

```
result <- conjoint(dataset = "mp3", dep_var = "Rating", indep_var = "Memory:Shape")
plot(result, scale_plot = TRUE)
plot(result, plots = "iw")
```

---

plot.correlation_	<i>Plot method for the correlation function</i>
-------------------	-------------------------------------------------

---

**Description**

Plot method for the correlation function

**Usage**

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



**Arguments**

x                      Return value from [correlation](#)

...                    further arguments passed to or from other methods.

**Details**

See <http://vnijs.github.io/radiant/quant/correlation.html> for an example in Radiant

**See Also**

[correlation](#) to calculate results

[summary.correlation\\_](#) to summarize results

**Examples**

```
result <- correlation("diamonds",c("price","carat","clarity"))
plot(result)
diamonds %>% correlation("price:clarity") %>% plot
```

---

plot.cross_tabs	<i>Plot method for the cross_tabs function</i>
-----------------	------------------------------------------------

---

**Description**

Plot method for the cross\_tabs function

**Usage**

```
## S3 method for class 'cross_tabs'
plot(x, check = "", shiny = FALSE, ...)
```

**Arguments**

x                      Return value from [cross\\_tabs](#)

check                  Show plots for variables var1 and var2. "observed" for the observed frequencies table, "expected" for the expected frequencies table (i.e., frequencies that would be expected if the null hypothesis holds), "chi\_sq" for the contribution to the overall chi-squared statistic for each cell (i.e.,  $(o - e)^2 / e$ ), "dev\_std" for the standardized differences between the observed and expected frequencies (i.e.,  $(o - e) / \sqrt{e}$ ), and "dev\_perc" for the percentage difference between the observed and expected frequencies (i.e.,  $(o - e) / e$ )

shiny                  Did the function call originate inside a shiny app

...                    further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/cross\\_tabs.html](http://vnijs.github.io/radiant/quant/cross_tabs.html) for an example in Radiant

**See Also**

[cross\\_tabs](#) to calculate results

[summary.cross\\_tabs](#) to summarize results

**Examples**

```
result <- cross_tabs("newspaper", "Income", "Newspaper")
plot(result, check = c("observed", "expected", "chi_sq"))
newspaper %>% cross_tabs("Income", "Newspaper") %>% plot(c("observed", "expected"))
```

---

plot.dtree	<i>Plot method for the dtree function</i>
------------	-------------------------------------------

---

**Description**

Plot method for the dtree function

**Usage**

```
## S3 method for class 'dtree'
plot(x, final = FALSE, shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">dtree</a>
final	If TRUE plot the decision tree solution, else the initial decision tree
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/quant/dtree.html> for an example in Radiant

**See Also**

[dtree](#) to generate the result

[summary.dtree](#) to summarize results

---

plot.full_factor	<i>Plot method for the full_factor function</i>
------------------	-------------------------------------------------

---

### Description

Plot method for the full\_factor function

### Usage

```
## S3 method for class 'full_factor'  
plot(x, shiny = FALSE, ...)
```

### Arguments

x	Return value from <code>full_factor</code>
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

### Details

See [http://vnijs.github.io/radiant/marketing/full\\_factor.html](http://vnijs.github.io/radiant/marketing/full_factor.html) for an example in Radiant

### See Also

`full_factor` to calculate results  
`plot.full_factor` to plot results

### Examples

```
result <- full_factor("diamonds",c("price","carat","table"))  
plot(result)  
result <- full_factor("computer","high_end:business")  
summary(result)
```

---

plot.glm_predict	<i>Plot method for the predict.glm_reg function</i>
------------------	-----------------------------------------------------

---

### Description

Plot method for the predict.glm\_reg function

### Usage

```
## S3 method for class 'glm_predict'  
plot(x, xvar = "", facet_row = ".", facet_col = ".",  
      color = "none", conf_lev = 0.95, ...)
```

**Arguments**

x	Return value from <code>predict.glm_reg</code> .
xvar	Variable to display along the X-axis of the plot
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 colour
conf_lev	Confidence level to use for prediction intervals (.95 is the default). Note that the error bars for predictions are approximations at this point.
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/glm\\_reg.html](http://vnijs.github.io/radiant/quant/glm_reg.html) for an example in Radiant

**See Also**

`glm_reg` to generate the result  
`summary.glm_reg` to summarize results  
`plot.glm_reg` to plot results  
`predict.glm_reg` to generate predictions

**Examples**

```
result <- glm_reg("titanic", "survived", c("pclass","sex","age"), lev = "Yes")
pred <- predict(result, pred_cmd = "pclass = levels(pclass)")
plot(pred, xvar = "pclass")
pred <- predict(result, pred_cmd = "age = 0:100")
plot(pred, xvar = "age")
pred <- predict(result, pred_cmd = "pclass = levels(pclass), sex = levels(sex)")
plot(pred, xvar = "pclass", color = "sex")
pred <- predict(result, pred_cmd = "pclass = levels(pclass), age = seq(0,100,20)")
plot(pred, xvar = "pclass", color = "age")
plot(pred, xvar = "age", color = "pclass")
pred <- predict(result, pred_cmd="pclass=levels(pclass), sex=levels(sex), age=seq(0,100,20)")
plot(pred, xvar = "age", color = "sex", facet_col = "pclass")
plot(pred, xvar = "age", color = "pclass", facet_col = "sex")
pred <- predict(result, pred_cmd="pclass=levels(pclass), sex=levels(sex), age=seq(0,100,5)")
plot(pred, xvar = "age", color = "sex", facet_col = "pclass")
plot(pred, xvar = "age", color = "pclass", facet_col = "sex")
```

---

plot.glm\_reg

---

*Plot method for the glm\_reg function*


---

**Description**

Plot method for the `glm_reg` function

**Usage**

```
## S3 method for class 'glm_reg'
plot(x, plots = "", conf_lev = 0.95, intercept = FALSE,
     shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">glm_reg</a>
plots	Plots to produce for the specified GLM model. Use "" to avoid showing any plots (default). "hist" shows histograms of all variables in the model. "scatter" shows scatter plots (or box plots for factors) for the dependent variable with each independent variable. "dashboard" is a series of four plots used to visually evaluate model. "coef" provides a coefficient plot
conf_lev	Confidence level to use for coefficient and odds confidence intervals (.95 is the default)
intercept	Include the intercept in the coefficient plot (TRUE or FALSE). FALSE is the default
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/glm\\_reg.html](http://vnijs.github.io/radiant/quant/glm_reg.html) for an example in Radiant

**See Also**

[glm\\_reg](#) to generate results  
[plot.glm\\_reg](#) to plot results  
[predict.glm\\_reg](#) to generate predictions  
[plot.glm\\_predict](#) to plot prediction output

**Examples**

```
result <- glm_reg("titanic", "survived", c("pclass","sex"), lev = "Yes")
plot(result, plots = "coef")
```

---

plot.hier\_clus

*Plot method for the hier\_clus function*


---

**Description**

Plot method for the hier\_clus function

**Usage**

```
## S3 method for class 'hier_clus'
plot(x, plots = c("scree", "diff"), cutoff = 0.02,
     shiny = TRUE, ...)
```

**Arguments**

x	Return value from <a href="#">hier_clus</a>
plots	Plots to return. "diff" shows the percentage change in within-cluster heterogeneity as respondents are group into different number of clusters, "dendro" shows the dendrogram, "scree" shows a scree plot of within-cluster heterogeneity
cutoff	For large datasets plots can take time to render and become hard to interpret. By selection a cutoff point (e.g., 0.05 percent) the initial steps in hierachical cluster analysis are removed from the plot
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/marketing/hier\\_clus.html](http://vnijs.github.io/radiant/marketing/hier_clus.html) for an example in Radiant

**See Also**

[summary.hier\\_clus](#) to summarize results

[plot.hier\\_clus](#) to plot results

**Examples**

```
result <- hier_clus("shopping", vars = c("v1:v6"))
plot(result, plots = c("diff", "scree"), cutoff = .05)
plot(result, plots = "dendro", cutoff = 0)
shopping %>% hier_clus(vars = c("v1:v6")) %>% plot
```

---

plot.kmeans_clus	<i>Plot method for kmeans_clus</i>
------------------	------------------------------------

---

**Description**

Plot method for kmeans\_clus

**Usage**

```
## S3 method for class 'kmeans_clus'
plot(x, shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">kmeans_clus</a>
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/marketing/kmeans\\_clus.html](http://vnijs.github.io/radiant/marketing/kmeans_clus.html) for an example in Radiant

**See Also**

[kmeans\\_clus](#) to generate results  
[summary.kmeans\\_clus](#) to summarize results  
[save\\_membership](#) to add cluster membership to the selected dataset

**Examples**

```
result <- kmeans_clus("shopping", vars = c("v1:v6"))  
plot(result)
```

---

plot.mds

*Plot method for the mds function*

---

**Description**

Plot method for the mds function

**Usage**

```
## S3 method for class 'mds'  
plot(x, rev_dim = "", fontsz = 1.3, ...)
```

**Arguments**

x	Return value from <a href="#">mds</a>
rev_dim	Flip the axes in plots
fontsz	Font size to use in plots
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/marketing/mds.html> for an example in Radiant

**See Also**

[mds](#) to calculate results  
[summary.mds](#) to plot results

**Examples**

```
result <- mds("city","from","to","distance")  
plot(result)  
plot(result, rev_dim = 1:2)  
plot(result, rev_dim = 1:2, fontsz = 2)
```

---

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, ...)
```

### 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
...	further arguments passed to or from other methods

### Details

See <http://vnijs.github.io/radiant/base/pivotr> for an example in Radiant

### See Also

[pivotr](#) to generate summaries  
[summary.pivotr](#) to show summaries

---

plot.pmap	<i>Plot method for the pmap function</i>
-----------	------------------------------------------

---

### Description

Plot method for the pmap function

### Usage

```
## S3 method for class 'pmap'  
plot(x, plots = "", scaling = 2.1, fontsz = 1.3, ...)
```



**Arguments**

x	Return value from <a href="#">pmap</a>
plots	Components to include in the plot ("brand", "attr"). If data on preferences is available use "pref" to add preference arrows to the plot
scaling	Arrow scaling in the brand map
fontsz	Font size to use in plots
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/marketing/pmap.html> for an example in Radiant

**See Also**

[pmap](#) to calculate results  
[summary.pmap](#) to plot results

**Examples**

```
result <- pmap("computer", "brand", "high_end:business")
plot(result, plots = "brand")
plot(result, plots = c("brand", "attr"))
plot(result, plots = c("brand", "attr"))
plot(result, scaling = 1, plots = c("brand", "attr"))
result <- pmap("computer", "brand", "high_end:dated",
               pref = c("innovative", "business"))
plot(result, plots = c("brand", "attr", "pref"))
```

---

plot.pre_factor	<i>Plot method for the pre_factor function</i>
-----------------	------------------------------------------------

---

**Description**

Plot method for the pre\_factor function

**Usage**

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

**Arguments**

x	Return value from <a href="#">pre_factor</a>
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/marketing/pre\\_factor.html](http://vnijs.github.io/radiant/marketing/pre_factor.html) for an example in Radiant

**See Also**

[pre\\_factor](#) to calculate results

[summary.pre\\_factor](#) to summarize results

**Examples**

```
result <- pre_factor("diamonds",c("price","carat","table"))
plot(result)
```

---

plot.regression	<i>Plot method for the regression function</i>
-----------------	------------------------------------------------

---

**Description**

Plot method for the regression function

**Usage**

```
## S3 method for class 'regression'
plot(x, plots = "", lines = "", conf_lev = 0.95,
     intercept = FALSE, shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">regression</a>
plots	Regression plots to produce for the specified regression model. Enter "" to avoid showing any plots (default). "hist" to show histograms of all variables in the model. "correlations" for a visual representation of the correlation matrix selected variables. "scatter" to show scatter plots (or box plots for factors) for the dependent variables with each independent variable. "dashboard" for a series of six plots that can be used to evaluate model fit visually. "resid_pred" to plot the independent variables against the model residuals. "coef" for a coefficient plot with adjustable confidence intervals. "leverage" to show leverage plots for each independent variable
lines	Optional lines to include in the select plot. "line" to include a line through a scatter plot. "loess" to include a polynomial regression fit line. To include both use c("line","loess")
conf_lev	Confidence level used to estimate confidence intervals (.95 is the default)
intercept	Include the intercept in the coefficient plot (TRUE, FALSE). FALSE is the default
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

**See Also**

[regression](#) to generate the results  
[summary.regression](#) to summarize results  
[predict.regression](#) to generate predictions

**Examples**

```
result <- regression("diamonds", "price", c("carat","clarity"))
plot(result, plots = "dashboard")
plot(result, plots = "dashboard", lines = c("line","loess"))
plot(result, plots = "coef", intercept = TRUE)
plot(result, plots = "coef", conf_lev = .99, intercept = TRUE)
plot(result, plots = "hist")
plot(result, plots = "scatter", lines = c("line","loess"))
plot(result, plots = "correlations")
plot(result, plots = "leverage")
plot(result, plots = "resid_pred", lines = "line")
```

---

plot.reg_predict	<i>Plot method for the predict.regression function</i>
------------------	--------------------------------------------------------

---

**Description**

Plot method for the predict.regression function

**Usage**

```
## S3 method for class 'reg_predict'
plot(x, xvar = "", facet_row = ".", facet_col = ".",
     color = "none", conf_lev = 0.95, ...)
```

**Arguments**

x	Return value from <a href="#">predict.regression</a> .
xvar	Variable to display along the X-axis of the plot
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 colour
conf_lev	Confidence level to use for prediction intervals (.95 is the default). Note that the error bars for predictions are approximations at this point.
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

**See Also**

[regression](#) to generate the result

[summary.regression](#) to summarize results

[plot.regression](#) to plot results

[predict.regression](#) to generate predictions

**Examples**

```
result <- regression("diamonds", "price", c("carat","clarity"))
pred <- predict(result, pred_cmd = "carat = 1:10")
plot(pred, xvar = "carat")
result <- regression("diamonds", "price", c("carat","clarity"), int_var = "carat:clarity")
dpred <- getdata("diamonds") %>% slice(1:100)
pred <- predict(result, pred_data = "dpred")
plot(pred, xvar = "carat", color = "clarity")
rm(dpred, envir = .GlobalEnv)
```

---

plot.repeater

---

*Plot repeated simulation*


---

**Description**

Plot repeated simulation

**Usage**

```
## S3 method for class 'repeater'
plot(x, sum_vars = "", byvar = "sim", fun = c("sum_rm",
  "mean_rm", "sd_rm"), shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">repeater</a>
sum_vars	(Numerical) variables to summaries
byvar	Variable(s) to group data by before summarizing
fun	Functions to use for summarizing
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

---

plot.simulator	<i>Plot method for the simulator function</i>
----------------	-----------------------------------------------

---

### Description

Plot method for the simulator function

### Usage

```
## S3 method for class 'simulator'  
plot(x, shiny = FALSE, ...)
```

### Arguments

x	Return value from <a href="#">simulator</a>
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

### Details

See <http://vnijs.github.io/radiant/quant/simulator> for an example in Radiant

### See Also

[single\\_mean](#) to generate the result  
[summary.single\\_mean](#) to summarize results

### Examples

```
result <- simulator(const = "cost 3", norm = "demand 2000 1000",  
                   discrete = "price 5 .3 8 .7",  
                   form = "profit = demand * (price - cost)")  
plot(result)
```

---

plot.single_mean	<i>Plot method for the single_mean function</i>
------------------	-------------------------------------------------

---

### Description

Plot method for the single\_mean function

### Usage

```
## S3 method for class 'single_mean'  
plot(x, plots = "hist", shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">single_mean</a>
plots	Plots to generate. "hist" shows a histogram of the data along with vertical lines that indicate the sample mean and the confidence interval. "simulate" shows the location of the sample mean and the comparison value (comp_value). Simulation is used to demonstrate the sampling variability in the data under the null-hypothesis
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/single\\_mean.html](http://vnijs.github.io/radiant/quant/single_mean.html) for an example in Radiant

**See Also**

[single\\_mean](#) to generate the result  
[summary.single\\_mean](#) to summarize results

**Examples**

```
result <- single_mean("diamonds", "price", comp_value = 3500)
plot(result, plots = c("hist", "simulate"))
```

---

plot.single_prop	<i>Plot method for the single_prop function</i>
------------------	-------------------------------------------------

---

**Description**

Plot method for the single\_prop function

**Usage**

```
## S3 method for class 'single_prop'
plot(x, plots = "hist", shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">single_prop</a>
plots	Plots to generate. "hist" shows a histogram of the data along with vertical lines that indicate the sample proportion and the confidence interval. "simulate" shows the location of the sample proportion and the comparison value (comp_value). Simulation is used to demonstrate the sampling variability in the data under the null-hypothesis
shiny	Did the function call originate inside a shiny app
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/single\\_prop.html](http://vnijs.github.io/radiant/quant/single_prop.html) for an example in Radiant

**See Also**

`single_prop` to generate the result  
`summary.single_prop` to summarize the results

**Examples**

```
result <- single_prop("diamonds", "clarity", lev = "IF", comp_value = 0.05)
plot(result, plots = c("hist", "simulate"))
result <- single_prop("titanic", "pclass", lev = "1st")
plot(result, plots = c("hist", "simulate"))
```

---

pmap	<i>Attribute based brand maps</i>
------	-----------------------------------

---

**Description**

Attribute based brand maps

**Usage**

```
pmap(dataset, brand, attr, pref = "", nr_dim = 2, 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
brand	A character variable with brand names
attr	Names of numeric variables
pref	Names of numeric brand preference measures
nr_dim	Number of dimensions
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")

**Details**

See <http://vnijs.github.io/radiant/marketing/pmap.html> for an example in Radiant

**Value**

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

**See Also**

`summary.pmap` to summarize results  
`plot.pmap` to plot results

**Examples**

```
result <- pmap("computer", "brand", "high_end:business")
```

---

predict.glm_reg	<i>Predict method for the glm_reg function</i>
-----------------	------------------------------------------------

---

## Description

Predict method for the glm\_reg function

## Usage

```
## S3 method for class 'glm_reg'
predict(object, pred_vars = "", pred_data = "",
        pred_cmd = "", prn = TRUE, ...)
```

## Arguments

object	Return value from <a href="#">glm_reg</a>
pred_vars	Variables selected to generate predictions
pred_data	Provide the name of a dataframe to generate predictions (e.g., "titanic"). The dataset must contain all columns used in the estimation
pred_cmd	Generate predictions using a command. For example, 'pclass = levels(pclass)' would produce predictions for the different levels of factor 'pclass'. To add another variable use a ',' (e.g., 'pclass = levels(pclass), age = seq(0,100,20)')
prn	Print prediction results (default is TRUE)
...	further arguments passed to or from other methods

## Details

See [http://vnijs.github.io/radiant/quant/glm\\_reg.html](http://vnijs.github.io/radiant/quant/glm_reg.html) for an example in Radiant

## See Also

[glm\\_reg](#) to generate the result  
[summary.glm\\_reg](#) to summarize results  
[plot.glm\\_reg](#) to plot results  
[plot.glm\\_predict](#) to plot prediction output

## Examples

```
result <- glm_reg("titanic", "survived", c("pclass","sex"), lev = "Yes")
predict(result, pred_cmd = "pclass = levels(pclass)")
glm_reg("titanic", "survived", c("pclass","sex"), lev = "Yes") %>%
  predict(pred_cmd = "sex = c('male','female')")
glm_reg("titanic", "survived", c("pclass","sex"), lev = "Yes") %>%
  predict(pred_data = "titanic")
```



---

predict.regression	<i>Predict method for the regression function</i>
--------------------	---------------------------------------------------

---

## Description

Predict method for the regression function

## Usage

```
## S3 method for class 'regression'
predict(object, pred_vars = "", pred_data = "",
        pred_cmd = "", conf_lev = 0.95, prn = TRUE, ...)
```

## Arguments

object	Return value from <a href="#">regression</a>
pred_vars	Variables to use for prediction
pred_data	Name of the dataset to use for prediction
pred_cmd	Command used to generate data for prediction
conf_lev	Confidence level used to estimate confidence intervals (.95 is the default)
prn	Print prediction results (default is TRUE)
...	further arguments passed to or from other methods

## Details

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

## See Also

[regression](#) to generate the result  
[summary.regression](#) to summarize results  
[plot.regression](#) to plot results

## Examples

```
result <- regression("diamonds", "price", c("carat","clarity"))
predict(result, pred_cmd = "carat = 1:10")
predict(result, pred_cmd = "clarity = levels(clarity)")
result <- regression("diamonds", "price", c("carat","clarity"), int_var = c("carat:clarity"))
dpred <- getdata("diamonds") %>% slice(1:10)
predict(result, pred_data = "dpred")
rm(dpred, envir = .GlobalEnv)
```

---

pre_factor	<i>Evaluate if data are appropriate for PCA / Factor analysis</i>
------------	-------------------------------------------------------------------

---

**Description**

Evaluate if data are appropriate for PCA / Factor analysis

**Usage**

```
pre_factor(dataset, vars, 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
vars	Variables to include in the analysis
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")

**Details**

See [http://vnijs.github.io/radiant/marketing/pre\\_factor.html](http://vnijs.github.io/radiant/marketing/pre_factor.html) for an example in Radiant

**Value**

A list with all variables defined in the function as an object of class `pre_factor`

**See Also**

`summary.pre_factor` to summarize results  
`plot.pre_factor` to plot results

**Examples**

```
result <- pre_factor("diamonds",c("price","carat","table"))
```

---

print.gtable	<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**

x                      a gtable object  
 ...                    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

---

publishers	<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")

---

radiant	<i>radiant</i>
---------	----------------

---

**Description**

radiant  
 Launch Radiant in the default browser

**Usage**

```
radiant(app = c("analytics", "marketing", "quant", "base"))
```

**Arguments**

app                    Choose the app to run. One of "base", "quant", "analytics", "marketing". "analytics" is the default

## Details

See <http://vnijs.github.io/radiant> for documentation and tutorials

## Examples

```
if (interactive()) {
  radiant("base")
  radiant("quant")
  radiant("marketing")
  radiant("analytics")
}
```

---

regression	<i>Linear regression using OLS</i>
------------	------------------------------------

---

## Description

Linear regression using OLS

## Usage

```
regression(dataset, dep_var, indep_var, int_var = "", check = "",
  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
dep_var	The dependent variable in the regression
indep_var	Independent variables in the regression
int_var	Interaction terms to include in the model
check	"standardize" to see standardized coefficient estimates. "stepwise" to apply stepwise selection of variables in estimation
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")

## Details

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

## Value

A list of all variables used in regression as an object of class `regression`

## See Also

[summary.regression](#) to summarize results  
[plot.regression](#) to plot results  
[predict.regression](#) to generate predictions

**Examples**

```
result <- regression("diamonds", "price", c("carat","clarity"))
result <- regression("diamonds", "price", c("carat","clarity"), check = "standardize")
```

---

repeater	<i>Repeat simulation</i>
----------	--------------------------

---

**Description**

Repeat simulation

**Usage**

```
repeater(nr = 12, vars = "", grid = "", seed = "", name = "",
  sim = "")
```

**Arguments**

nr	Number times to repeat the simulation
vars	Variables to use in repeated simulation
grid	Expression to use in grid search for constants
seed	To repeat a simulation with the same randomly generated values enter a number into Random seed input box.
name	To save the simulated data for further analysis specify a name in the Sim name input box. You can then investigate the simulated data by choosing the specified name from the Datasets dropdown in any of the other Data tabs.
sim	Return value from the simulator function

**Examples**

```
result <- simulator(const = "cost 3", norm = "demand 2000 1000",
  discrete = "price 5 .3 8 .7",
  form = "profit = demand * (price - cost)")

repeater(sim = result)
```

---

rndnames	<i>100 random names</i>
----------	-------------------------

---

**Description**

100 random names

**Usage**

```
data(rndnames)
```

**Format**

A data frame with 100 rows and 2 variables

**Details**

A list of 100 random names generated by [listofrandomnames.com](http://listofrandomnames.com). Description provided in `attr(rndnames,"description")`

---

sample_size	<i>Sample size calculation</i>
-------------	--------------------------------

---

**Description**

Sample size calculation

**Usage**

```
sample_size(type = "mean", err_mean = 2, sd_mean = 10, err_prop = 0.1,
  p_prop = 0.5, zval = 1.96, incidence = 1, response = 1,
  pop_correction = "no", pop_size = 1000000)
```

**Arguments**

type	Choose "mean" or "proportion"
err_mean	Acceptable Error for Mean
sd_mean	Standard deviation for Mean
err_prop	Acceptable Error for Proportion
p_prop	Initial proportion estimate for Proportion
zval	Z-value
incidence	Incidence rate (i.e., fraction of valid respondents)
response	Response rate
pop_correction	Apply correction for population size ("yes","no")
pop_size	Population size

**Details**

See [http://vnijs.github.io/radiant/quant/sample\\_size.html](http://vnijs.github.io/radiant/quant/sample_size.html) for an example in Radiant

**Value**

A list of variables defined in sample\_size as an object of class sample\_size

**See Also**

[summary.sample\\_size](#) to summarize results

**Examples**

```
result <- sample_size(type = "mean", err_mean = 2, sd_mean = 10)
```

sampling

*Simple random sampling***Description**

Simple random sampling

**Usage**

```
sampling(dataset, var, sample_size, 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
var	The variable to sample from
sample_size	Number of units to select
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")

**Details**

See <http://vnijs.github.io/radiant/quant/sampling.html> for an example in Radiant

**Value**

A list of variables defined in sampling as an object of class `sampling`

**See Also**

[summary.sampling](#) to summarize results

**Examples**

```
result <- sampling("rndnames", "Names", 10)
```

save\_factors

*Save factor scores to active dataset***Description**

Save factor scores to active dataset

**Usage**

```
save_factors(object)
```

**Arguments**

object	Return value from <a href="#">full_factor</a>
--------	-----------------------------------------------

## Details

See [http://vnijs.github.io/radiant/marketing/full\\_factor.html](http://vnijs.github.io/radiant/marketing/full_factor.html) for an example in Radiant

## Examples

```
result <- full_factor("diamonds", c("price", "carat", "table"))
save_factors(result)
head(diamonds)
```

---

save_membership	<i>Add a cluster membership variable to the active dataset</i>
-----------------	----------------------------------------------------------------

---

## Description

Add a cluster membership variable to the active dataset

## Usage

```
save_membership(object)
```

## Arguments

object	Return value from <a href="#">kmeans_clus</a>
--------	-----------------------------------------------

## Details

See [http://vnijs.github.io/radiant/marketing/kmeans\\_clus.html](http://vnijs.github.io/radiant/marketing/kmeans_clus.html) for an example in Radiant

## See Also

[kmeans\\_clus](#) to generate results  
[summary.kmeans\\_clus](#) to summarize results  
[plot.kmeans\\_clus](#) to plot results

## Examples

```
result <- kmeans_clus("shopping", vars = c("v1:v6"))
save_membership(result)
head(shopping)
```



---

`sd_rm`*Standard deviation with na.rm = TRUE*

---

**Description**

Standard deviation with na.rm = TRUE

**Usage**

```
sd_rm(x)
```

**Arguments**

`x`                      Input variable

**Value**

Standard deviation

**Examples**

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

---

`serr`*Standard error*

---

**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_class	<i>Alias used to set the class for analysis function return</i>
-----------	-----------------------------------------------------------------

---

**Description**

Alias used to set the class for analysis function return

**Usage**

```
set_class()
```

**Examples**

```
foo <- function(x) x^2 %>% set_class(c("foo", class(.)))
```

---

shopping	<i>Shopping attitudes</i>
----------	---------------------------

---

**Description**

Shopping attitudes

**Usage**

```
data(shopping)
```

**Format**

A data frame with 20 rows and 7 variables

**Details**

Attitudinal data on shopping for 20 consumers. Description provided in attr(shopping,"description")

---

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))
```

---

simulater	<i>Simulate data for decision analysis</i>
-----------	--------------------------------------------

---

**Description**

Simulate data for decision analysis

**Usage**

```
simulater(const = "", norm = "", unif = "", discrete = "", form = "",
  seed = "", name = "", nr = 1000, dat = NULL)
```

**Arguments**

const	A string listing the constants to include in the analysis (e.g., "cost = 3; size = 4")
norm	A string listing the normally distributed random variables to include in the analysis (e.g., "demand 2000 1000" where the first number is the mean and the second is the standard deviation)
unif	A string listing the uniformly distributed random variables to include in the analysis (e.g., "demand 0 1" where the first number is the minimum value and the second is the maximum value)
discrete	A string listing the random variables with a discrete distribution to include in the analysis (e.g., "price 5 .3 8 .7" where for each pair of numbers the first is the value and the second the probability)
form	A string with the formula to evaluate (e.g., "profit = demand * (price - cost)")
seed	To repeat a simulation with the same randomly generated values enter a number into Random seed input box.
name	To save the simulated data for further analysis specify a name in the Sim name input box. You can then investigate the simulated data by choosing the specified name from the Datasets dropdown in any of the other Data tabs.
nr	Number of simulation runs
dat	Data list from previous simulation. Used by repeater function

**Details**

See <http://vnijs.github.io/radiant/quant/simulater.html> for an example in Radiant

**Value**

A data.frame with the created variables

**See Also**

[summary.simulater](#) to summarize results

[plot.simulater](#) to plot results

**Examples**

```
result <- simulater(const = "cost 3", norm = "demand 2000 1000",
  discrete = "price 5 .3 8 .7",
  form = "profit = demand * (price - cost)")
```

---

single_mean	<i>Compare a sample mean to a population mean</i>
-------------	---------------------------------------------------

---

**Description**

Compare a sample mean to a population mean

**Usage**

```
single_mean(dataset, var, comp_value = 0, alternative = "two.sided",
  conf_lev = 0.95, 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
var	The variable selected for the mean comparison
comp_value	Population value to compare to the sample mean
alternative	The alternative hypothesis ("two.sided", "greater", or "less")
conf_lev	Span for the confidence interval
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")

**Details**

See [http://vnijs.github.io/radiant/quant/single\\_mean.html](http://vnijs.github.io/radiant/quant/single_mean.html) for an example in Radiant

**Value**

A list of variables defined in `single_mean` as an object of class `single_mean`

**See Also**

[summary.single\\_mean](#) to summarize results

[plot.single\\_mean](#) to plot results

**Examples**

```
single_mean("diamonds", "price")
```

---

single\_prop

---

*Compare a sample proportion to a population proportion*


---

**Description**

Compare a sample proportion to a population proportion

**Usage**

```
single_prop(dataset, var, lev = "", comp_value = 0.5,
  alternative = "two.sided", conf_lev = 0.95, 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
var	The variable selected for the proportion comparison
lev	The factor level selected for the proportion comparison
comp_value	Population value to compare to the sample proportion
alternative	The alternative hypothesis ("two.sided", "greater", or "less")
conf_lev	Span of the confidence interval
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")

**Details**

See [http://vnijs.github.io/radiant/quant/single\\_prop.html](http://vnijs.github.io/radiant/quant/single_prop.html) for an example in Radiant

**Value**

A list of variables used in `single_prop` as an object of class `single_prop`

**See Also**

`summary.single_prop` to summarize the results

`plot.single_prop` to plot the results

**Examples**

```
result <- single_prop("diamonds", "cut")
result <- single_prop("diamonds", "clarity", lev = "IF", comp_value = 0.05)
```

---

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

---

**Description**

Exporting the skew function from the psych package

---

ssh	<i>Hide warnings and messages and return invisible</i>
-----	--------------------------------------------------------

---

**Description**

Hide warnings and messages and return invisible

**Usage**

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

**Arguments**

...                      Inputs to keep quiete

**Details**

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

**Examples**

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

---

sshhr	<i>Hide warnings and messages and return result</i>
-------	-----------------------------------------------------

---

**Description**

Hide warnings and messages and return result

**Usage**

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

**Arguments**

...                      Inputs to keep quiete

**Details**

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

**Examples**

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

---

state_init	<i>Set initial value for shiny input</i>
------------	------------------------------------------

---

## Description

Set initial value for shiny input

## Usage

```
state_init(inputvar, init = "")
```

## Arguments

inputvar	Name shiny input
init	Initial value to use if state value for input not set

## Details

Useful for radio button or checkbox

## Value

value for inputvar

## See Also

[state\\_single](#)  
[state\\_multiple](#)  
[copy\\_from](#)

## Examples

```
r_state <- list()
state_init("test")
state_init("test",0)
r_state$test <- c("a","b")
state_init("test",0)
shiny::radioButtons("rb", label = "Button:", c("a","b"), selected = state_init("rb", "a"))
r_state$rb <- "b"
shiny::radioButtons("rb", label = "Button:", c("a","b"), selected = state_init("rb", "a"))
rm(r_state)
```

---

state_multiple	<i>Set initial values for shiny input from a list of values</i>
----------------	-----------------------------------------------------------------

---

### Description

Set initial values for shiny input from a list of values

### Usage

```
state_multiple(inputvar, vals, init = character(0))
```

### Arguments

inputvar	Name shiny input
vals	Possible values for inputvar
init	Initial value to use if state value for input not set

### Details

Useful for select input with multiple = TRUE and when you want to use inputs selected for another tool (e.g., pre\_factor and full\_factor or hier\_clus and kmeans\_clus in Radiant)

### Value

value for inputvar

### See Also

[state\\_init](#)  
[state\\_single](#)  
[copy\\_from](#)

### Examples

```
r_state <- list()
state_multiple("test",1:10,1:3)
r_state$test <- 8:10
state_multiple("test",1:10,1:3)
shiny::selectInput("sim", label = "Select:", c("a","b"),
  selected = state_multiple("sim", c("a","b")), multiple = TRUE)
r_state$sim <- c("a","b")
shiny::selectInput("sim", label = "Select:", c("a","b"),
  selected = state_single("sim", c("a","b")), multiple = TRUE)
```



---

state_single	<i>Set initial value for shiny input from a list of values</i>
--------------	----------------------------------------------------------------

---

## Description

Set initial value for shiny input from a list of values

## Usage

```
state_single(inputvar, vals, init = character(0))
```

## Arguments

inputvar	Name shiny input
vals	Possible values for inputvar
init	Initial value to use if state value for input not set

## Details

Useful for select input with multiple = FALSE

## Value

value for inputvar

## See Also

[state\\_init](#)  
[state\\_multiple](#)  
[copy\\_from](#)

## Examples

```
r_state <- list()
state_single("test",1:10,1)
r_state$test <- 8
state_single("test",1:10,1)
shiny::selectInput("si", label = "Select:", c("a","b"), selected = state_single("si"))
r_state$si <- "b"
shiny::selectInput("si", label = "Select:", c("a","b"), selected = state_single("si", "b"))
```

---

store_glm	<i>Store residuals or predicted values generated in the glm_reg function</i>
-----------	------------------------------------------------------------------------------

---

### Description

Store residuals or predicted values generated in the glm\_reg function

### Usage

```
store_glm(object, data = object$dataset, type = "residual",
  name = paste0(type, "_glm"))
```

### Arguments

object	Return value from <code>glm_reg</code> or <code>predict.glm_reg</code>
data	Dataset name
type	Residuals ("residual") or predictions ("predictions")
name	Variable name assigned to the residuals or predicted values

### Details

See [http://vnijs.github.io/radiant/quant/glm\\_reg.html](http://vnijs.github.io/radiant/quant/glm_reg.html) for an example in Radiant

### Examples

```
result <- glm_reg("titanic", "survived", "pclass", lev = "Yes")
store_glm(result, "Prediction")
head(titanic)
```

---

store_reg	<i>Store residuals or predicted values generated in the regression function</i>
-----------	---------------------------------------------------------------------------------

---

### Description

Store residuals or predicted values generated in the regression function

### Usage

```
store_reg(object, data = object$dataset, type = "residual",
  name = paste0(type, "_reg"))
```

### Arguments

object	Return value from <code>regression</code> or <code>predict.regression</code>
data	Dataset name
type	Residuals ("residual") or predictions ("predictions")
name	Variable name assigned to the residuals or predicted values

## Details

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

## Examples

```
result <- regression("diamonds", "price", c("carat","clarity"))
store_reg(result)
head(diamonds)
```

---

summary.compare\_means *Summary method for the compare\_means function*

---

## Description

Summary method for the compare\_means function

## Usage

```
## S3 method for class 'compare_means'
summary(object, ...)
```

## Arguments

object	Return value from <a href="#">compare_means</a>
...	further arguments passed to or from other methods

## Details

See [http://vnijs.github.io/radiant/quant/compare\\_means.html](http://vnijs.github.io/radiant/quant/compare_means.html) for an example in Radiant

## See Also

[compare\\_means](#) to calculate results  
[plot.compare\\_means](#) to plot results

## Examples

```
result <- compare_means("diamonds","cut","price")
summary(result)
result <- diamonds %>% tbl_df %>% compare_means("x","y")
summary(result)
result <- diamonds %>% tbl_df %>% group_by(cut) %>% compare_means("x",c("x","y"))
summary(result)
```

---

summary.compare\_props *Summary method for the compare\_props function*

---

### Description

Summary method for the compare\_props function

### Usage

```
## S3 method for class 'compare_props'
summary(object, ...)
```

### Arguments

object	Return value from <a href="#">compare_props</a>
...	further arguments passed to or from other methods

### Details

See [http://vnijs.github.io/radiant/quant/compare\\_props.html](http://vnijs.github.io/radiant/quant/compare_props.html) for an example in Radiant

### See Also

[compare\\_props](#) to calculate results  
[plot.compare\\_props](#) to plot results

### Examples

```
result <- compare_props("titanic", "pclass", "survived")
summary(result)
titanic %>% compare_props("pclass", "survived") %>% summary
```

---

summary.conjoint *Summary method for the conjoint function*

---

### Description

Summary method for the conjoint function

### Usage

```
## S3 method for class 'conjoint'
summary(object, mc_diag = FALSE, ...)
```

### Arguments

object	Return value from <a href="#">conjoint</a>
mc_diag	Shows multicollinearity diagnostics.
...	further arguments passed to or from other methods

## Details

See <http://vnijs.github.io/radiant/marketing/conjoint.html> for an example in Radiant

## See Also

[conjoint](#) to generate results

[plot.conjoint](#) to plot results

## Examples

```
result <- conjoint("mp3", dep_var = "Rating", indep_var = "Memory:Shape")
summary(result, mc_diag = TRUE)
mp3 %>% conjoint(dep_var = "Rating", indep_var = "Memory:Shape") %>% summary(., mc_diag = TRUE)
```

---

summary.conjoint\_profiles

*Summary method for the conjoint\_profiles function*

---

## Description

Summary method for the conjoint\_profiles function

## Usage

```
## S3 method for class 'conjoint_profiles'
summary(object, ...)
```

## Arguments

object	Return value from <a href="#">conjoint_profiles</a>
...	further arguments passed to or from other methods.

## Details

See [http://vnijs.github.io/radiant/marketing/conjoint\\_profiles.html](http://vnijs.github.io/radiant/marketing/conjoint_profiles.html) for an example in Radiant

## See Also

[conjoint\\_profiles](#) to calculate results

## Examples

```
cp <- readLines(system.file("examples/profiles-movie.txt", package='radiant'))
result <- conjoint_profiles("cp")
summary(result)
rm(cp, envir = .GlobalEnv)
readLines(system.file("examples/profiles-movie.txt", package='radiant')) %>%
  conjoint_profiles %>% summary
```

---

summary.correlation\_     *Summary method for the correlation function*

---

### Description

Summary method for the correlation function

### Usage

```
## S3 method for class 'correlation_'
summary(object, cutoff = 0, ...)
```

### Arguments

object	Return value from <a href="#">correlation</a>
cutoff	Show only correlations larger than the cutoff in absolute value. Default is a cutoff of 0
...	further arguments passed to or from other methods.

### Details

See <http://vnijs.github.io/radiant/quant/correlation.html> for an example in Radiant

### See Also

[correlation](#) to calculate results  
[plot.correlation\\_](#) to plot results

### Examples

```
result <- correlation("diamonds",c("price","carat","clarity"))
summary(result, cutoff = .3)
diamonds %>% correlation("price:clarity") %>% summary
```

---

summary.cross\_tabs     *Summary method for the cross\_tabs function*

---

### Description

Summary method for the cross\_tabs function

### Usage

```
## S3 method for class 'cross_tabs'
summary(object, check = "", ...)
```

**Arguments**

object	Return value from <a href="#">cross_tabs</a>
check	Show table(s) for variables var1 and var2. "observed" for the observed frequencies table, "expected" for the expected frequencies table (i.e., frequencies that would be expected if the null hypothesis holds), "chi_sq" for the contribution to the overall chi-squared statistic for each cell (i.e., $(o - e)^2 / e$ ), "dev_std" for the standardized differences between the observed and expected frequencies (i.e., $(o - e) / \sqrt{e}$ ), and "dev_perc" for the percentage difference between the observed and expected frequencies (i.e., $(o - e) / e$ )
...	further arguments passed to or from other methods.

**Details**

See [http://vnijs.github.io/radiant/quant/cross\\_tabs.html](http://vnijs.github.io/radiant/quant/cross_tabs.html) for an example in Radiant

**See Also**

[cross\\_tabs](#) to calculate results  
[plot.cross\\_tabs](#) to plot results

**Examples**

```
result <- cross_tabs("newspaper", "Income", "Newspaper")
summary(result, check = c("observed", "expected", "chi_sq"))
newspaper %>% cross_tabs("Income", "Newspaper") %>% summary("observed")
```

---

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

---

**Description**

Summary method for the dree function

**Usage**

```
## S3 method for class 'dtree'
summary(object, ...)
```

**Arguments**

object	Return value from <a href="#">simulator</a>
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/quant/dtree.html> for an example in Radiant

**See Also**

[dtree](#) to generate the results  
[plot.dtree](#) to plot results

---

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", ...)
```

### Arguments

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

### Details

See <http://vnijs.github.io/radiant/base/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.full_factor	<i>Summary method for the full_factor function</i>
---------------------	----------------------------------------------------

---

### Description

Summary method for the full\_factor function

### Usage

```
## S3 method for class 'full_factor'
summary(object, cutoff = 0, fsort = FALSE, ...)
```



**Arguments**

object	Return value from <a href="#">full_factor</a>
cutoff	Show only loadings with (absolute) values above cutoff (default = 0)
fsort	Sort factor loadings
...	further arguments passed to or from other methods

**Details**

See [http://vnij.s.github.io/radiant/marketing/full\\_factor.html](http://vnij.s.github.io/radiant/marketing/full_factor.html) for an example in Radiant

**See Also**

[full\\_factor](#) to calculate results

[plot.full\\_factor](#) to plot results

**Examples**

```
result <- full_factor("diamonds",c("price","carat","depth","table","x"))
summary(result)
summary(result, cutoff = 0, fsort = FALSE)
summary(result, cutoff = 0, fsort = TRUE)
summary(result, cutoff = .5, fsort = TRUE)
diamonds %>% full_factor(c("price","carat","depth","table","x")) %>% summary
diamonds %>% full_factor(c("price","carat","depth","table","x")) %>% summary(cutoff = .5)
```

summary.glm\_reg

*Summary method for the glm\_reg function***Description**

Summary method for the glm\_reg function

**Usage**

```
## S3 method for class 'glm_reg'
summary(object, sum_check = "", conf_lev = 0.95,
        test_var = "", ...)
```

**Arguments**

object	Return value from <a href="#">glm_reg</a>
sum_check	Optional output or estimation parameters. "rsme" to show the root mean squared error. "sumsquares" to show the sum of squares table. "vif" to show multi-collinearity diagnostics. "confint" to show coefficient confidence interval estimates.
conf_lev	Confidence level to use for coefficient and odds confidence intervals (.95 is the default)
test_var	Variables to evaluate in model comparison (i.e., a competing models Chi-squared test)
...	further arguments passed to or from other methods

## Details

See [http://vnijs.github.io/radiant/quant/glm\\_reg.html](http://vnijs.github.io/radiant/quant/glm_reg.html) for an example in Radiant

## See Also

[glm\\_reg](#) to generate the results  
[plot.glm\\_reg](#) to plot the results  
[predict.glm\\_reg](#) to generate predictions  
[plot.glm\\_predict](#) to plot prediction output

## Examples

```
result <- glm_reg("titanic", "survived", "pclass", lev = "Yes")
summary(result, test_var = "pclass")
res <- glm_reg("titanic", "survived", c("pclass", "sex"), int_var="pclass:sex", lev="Yes")
summary(res, sum_check = c("vif", "confint", "odds"))
titanic %>% glm_reg("survived", c("pclass", "sex", "age"), lev = "Yes") %>% summary("vif")
```

---

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

---

## Description

Summary method for the hier\_clus function

## Usage

```
## S3 method for class 'hier_clus'
summary(object, ...)
```

## Arguments

object	Return value from <a href="#">hier_clus</a>
...	further arguments passed to or from other methods

## Details

See [http://vnijs.github.io/radiant/marketing/hier\\_clus.html](http://vnijs.github.io/radiant/marketing/hier_clus.html) for an example in Radiant

## See Also

[summary.hier\\_clus](#) to summarize results  
[plot.hier\\_clus](#) to plot results

## Examples

```
result <- hier_clus("shopping", vars = c("v1:v6"))
summary(result)
```

---

summary.kmeans_clus	<i>Summary method for kmeans_clus</i>
---------------------	---------------------------------------

---

### Description

Summary method for kmeans\_clus

### Usage

```
## S3 method for class 'kmeans_clus'  
summary(object, ...)
```

### Arguments

object	Return value from <a href="#">kmeans_clus</a>
...	further arguments passed to or from other methods

### Details

See [http://vnijs.github.io/radiant/marketing/kmeans\\_clus.html](http://vnijs.github.io/radiant/marketing/kmeans_clus.html) for an example in Radiant

### See Also

[kmeans\\_clus](#) to generate results  
[plot.kmeans\\_clus](#) to plot results  
[save\\_membership](#) to add cluster membership to the selected dataset

### Examples

```
result <- kmeans_clus("shopping", vars = c("v1:v6"))  
summary(result)  
shopping %>% kmeans_clus(vars = c("v1:v6"), nr_clus = 3) %>% summary
```

---

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

---

### Description

Summary method for the mds function

### Usage

```
## S3 method for class 'mds'  
summary(object, dec = 1, ...)
```

**Arguments**

object	Return value from <a href="#">mds</a>
dec	Rounding to use for output (default = 0). +1 used for coordinates. +2 used for stress measure. Not currently accessible in Radiant
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/marketing/mds.html> for an example in Radiant

**See Also**

[mds](#) to calculate results  
[plot.mds](#) to plot results

**Examples**

```
result <- mds("city", "from", "to", "distance")
summary(result)
summary(result, dec = 2)
city %>% mds("from", "to", "distance") %>% summary
```

---

summary.pivotr

*Summary method for pivotr*


---

**Description**

Summary method for pivotr

**Usage**

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

**Arguments**

object	Return value from <a href="#">pivotr</a>
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://vnijs.github.io/radiant/base/pivotr.html> for an example in Radiant

**See Also**

[pivotr](#) to create the pivot-table using dplyr

**Examples**

```

pivotr("diamonds", cvars = "cut") %>% summary
pivotr("diamonds", cvars = "cut") %>% summary
pivotr("diamonds", cvars = "cut:clarity", nvar = "price") %>% summary

```

summary.pmap

*Summary method for the pmap function***Description**

Summary method for the pmap function

**Usage**

```

## S3 method for class 'pmap'
summary(object, cutoff = 0, ...)

```

**Arguments**

object	Return value from <a href="#">pmap</a>
cutoff	Show only loadings with (absolute) values above cutoff (default = 0)
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/marketing/pmap.html> for an example in Radiant

**See Also**

[pmap](#) to calculate results

[plot.pmap](#) to plot results

**Examples**

```

result <- pmap("computer", "brand", "high_end:business")
summary(result)
summary(result, cutoff = .3)
result <- pmap("computer", "brand", "high_end:dated", pref = c("innovative", "business"))
summary(result)
computer %>% pmap("brand", "high_end:dated", pref = c("innovative", "business")) %>%
  summary

```

---

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

---

**Description**

Summary method for the pre\_factor function

**Usage**

```
## S3 method for class 'pre_factor'
summary(object, ...)
```

**Arguments**

object	Return value from <a href="#">pre_factor</a>
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/marketing/pre\\_factor.html](http://vnijs.github.io/radiant/marketing/pre_factor.html) for an example in Radiant

**See Also**

[pre\\_factor](#) to calculate results  
[plot.pre\\_factor](#) to plot results

**Examples**

```
result <- pre_factor("diamonds",c("price","carat","table"))
summary(result)
diamonds %>% pre_factor(c("price","carat","table")) %>% summary
result <- pre_factor("computer","high_end:business")
summary(result)
```

---

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

---

**Description**

Summary method for the regression function

**Usage**

```
## S3 method for class 'regression'
summary(object, sum_check = "", conf_lev = 0.95,
        test_var = "", ...)
```

**Arguments**

object	Return value from <a href="#">regression</a>
sum_check	Optional output or estimation parameters. "rmse" to show the root mean squared error. "sumsquares" to show the sum of squares table. "vif" to show multi-collinearity diagnostics. "confint" to show coefficient confidence interval estimates.
conf_lev	Confidence level used to estimate confidence intervals (.95 is the default)
test_var	Variables to evaluate in model comparison (i.e., a competing models F-test)
...	further arguments passed to or from other methods

**Details**

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

**See Also**

[regression](#) to generate the results  
[plot.regression](#) to plot results  
[predict.regression](#) to generate predictions

**Examples**

```
result <- regression("diamonds", "price", c("carat","clarity"))
summary(result, sum_check = c("rmse","sumsquares","vif","confint"), test_var = "clarity")
result <- regression("shopping", "v1", c("v2","v3"))
summary(result, test_var = "v2")
shopping %>% regression("v1", "v2:v6") %>% summary
```

---

summary.repeater	<i>Summarize repeated simulation</i>
------------------	--------------------------------------

---

**Description**

Summarize repeated simulation

**Usage**

```
## S3 method for class 'repeater'
summary(object, sum_vars = "", byvar = "",
  fun = c("sum_rm", "mean_rm", "sd_rm"), ...)
```

**Arguments**

object	Return value from <a href="#">repeater</a>
sum_vars	(Numerical) variables to summaries
byvar	Variable(s) to group data by before summarizing
fun	Functions to use for summarizing
...	further arguments passed to or from other methods

---

summary.sample_size	Summary method for the sample_size function
---------------------	---------------------------------------------

---

### Description

Summary method for the sample\_size function

### Usage

```
## S3 method for class 'sample_size'  
summary(object, ...)
```

### Arguments

object	Return value from <a href="#">sample_size</a>
...	further arguments passed to or from other methods

### Details

See [http://vnijs.github.io/radiant/quant/sample\\_size](http://vnijs.github.io/radiant/quant/sample_size) for an example in Radiant

### See Also

[sample\\_size](#) to generate the results

### Examples

```
result <- sample_size(type = "mean", err_mean = 2, sd_mean = 10)  
summary(result)
```

---

summary.sampling	Summary method for the sampling function
------------------	------------------------------------------

---

### Description

Summary method for the sampling function

### Usage

```
## S3 method for class 'sampling'  
summary(object, print_sf = TRUE, ...)
```

### Arguments

object	Return value from <a href="#">sampling</a>
print_sf	Print full sampling frame. Default is TRUE
...	further arguments passed to or from other methods



## Details

See <http://vnijs.github.io/radiant/quant/sampling> for an example in Radiant

## See Also

[sampling](#) to generate the results

## Examples

```
set.seed(1234)
result <- sampling("rndnames", "Names", 10)
summary(result)
```

---

summary.simulater

*Summary method for the simulater function*

---

## Description

Summary method for the simulater function

## Usage

```
## S3 method for class 'simulater'
summary(object, ...)
```

## Arguments

object	Return value from <a href="#">simulater</a>
...	further arguments passed to or from other methods

## Details

See <http://vnijs.github.io/radiant/quant/simulater.html> for an example in Radiant

## See Also

[simulater](#) to generate the results

[plot.simulater](#) to plot results

## Examples

```
result <- simulater(norm = "demand 2000 1000")
summary(result)
```

---

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

---

### Description

Summary method for the single\_mean function

### Usage

```
## S3 method for class 'single_mean'
summary(object, ...)
```

### Arguments

object	Return value from <a href="#">single_mean</a>
...	further arguments passed to or from other methods

### Details

See [http://vnijs.github.io/radiant/quant/single\\_mean.html](http://vnijs.github.io/radiant/quant/single_mean.html) for an example in Radiant

### See Also

[single\\_mean](#) to generate the results  
[plot.single\\_mean](#) to plot results

### Examples

```
result <- single_mean("diamonds", "price")
summary(result)
diamonds %>% single_mean("price") %>% summary
```

---

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

---

### Description

Summary method for the single\_prop function

### Usage

```
## S3 method for class 'single_prop'
summary(object, ...)
```

### Arguments

object	Return value from <a href="#">single_prop</a>
...	further arguments passed to or from other methods

**Details**

See [http://vnijs.github.io/radiant/quant/single\\_prop.html](http://vnijs.github.io/radiant/quant/single_prop.html) for an example in Radiant

**See Also**

`single_prop` to generate the results

`plot.single_prop` to plot the results

**Examples**

```
result <- single_prop("diamonds","clarity", lev = "IF", comp_value = 0.05)
summary(result)
diamonds %>% single_prop("clarity", lev = "IF", comp_value = 0.05) %>% summary
```

---

sum_rm	<i>Sum with na.rm = TRUE</i>
--------	------------------------------

---

**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	<i>Super heroes</i>
-------------	---------------------

---

**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")

---

test_specs	<i>Add interaction terms to list of test variables if needed</i>
------------	------------------------------------------------------------------

---

**Description**

Add interaction terms to list of test variables if needed

**Usage**

```
test_specs(test_var, int_var)
```

**Arguments**

test_var	List of variables to use for testing for regression or glm_reg
int_var	Interaction terms specified

**Details**

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

**Value**

A vector of variables names to test

**Examples**

```
test_specs("a", c("a:b", "b:c"))
```

---

the_table	<i>Function to calculate the PW and IW table for conjoint</i>
-----------	---------------------------------------------------------------

---

**Description**

Function to calculate the PW and IW table for conjoint

**Usage**

```
the_table(model, dat, indep_var)
```

**Arguments**

model	Tidied model results (broom) output from <a href="#">conjoint</a> passed on by summary.conjoint
dat	Conjoint data
indep_var	Independent variables used in the conjoint regression

**Details**

See <http://vnijs.github.io/radiant/marketing/conjoint.html> for an example in Radiant

**See Also**

`conjoint` to generate results  
`summary.conjoint` to summarize results  
`plot.conjoint` to plot results

**Examples**

```
result <- conjoint(dataset = "mp3", dep_var = "Rating", indep_var = "Memory:Shape")
the_table(result$model, result$dat, result$indep_var)
```

---

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

---

**Description**

Survival data for the Titanic

**Usage**

```
data(titanic)
```

**Format**

A data frame with 1309 rows and 11 variables

**Details**

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

---

titanic_pred	<i>Predict survival</i>
--------------	-------------------------

---

**Description**

Predict survival

**Usage**

```
data(titanic_pred)
```

**Format**

A data frame with 6 rows and 3 variables

**Details**

Prediction data.frame for `glm_reg` based on the Titanic dataset

toothpaste	<i>Toothpaste attitudes</i>
<b>Description</b> Toothpaste attitudes	
<b>Usage</b> <pre>data(toothpaste)</pre>	
<b>Format</b> A data frame with 60 rows and 10 variables	
<b>Details</b> Attitudinal data on toothpaste for 60 consumers. Description provided in attr(toothpaste,"description")	
var_check	<i>Check if main effects for all interaction effects are included in the model If ':' is used to select a range _indep_var_ is updated</i>

### Description

Check if main effects for all interaction effects are included in the model If ':' is used to select a range \_indep\_var\_ is updated

### Usage

```
var_check(iv, cn, intv = "")
```

### Arguments

iv	List of independent variables provided to _regression_ or _glm_
cn	Column names for all independent variables in _dat_
intv	Interaction terms specified

### Details

See <http://vnijs.github.io/radiant/quant/regression.html> for an example in Radiant

### Value

'vars' is a vector of right-hand side variables, possibly with interactions, 'iv' is the list of independent variables, and intv are interaction terms

### Examples

```
var_check("a:d", c("a", "b", "c", "d"))
var_check(c("a", "b"), c("a", "b"), "a:c")
```

---

viewdata	<i>View data</i>
----------	------------------

---

## Description

View data

## Usage

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

## Arguments

dataset	Name of the dataframe to change
vars	Variables to so (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	<i>Visualize data using ggplot2</i> <a href="http://docs.ggplot2.org/current/">http://docs.ggplot2.org/current/</a>
-----------	---------------------------------------------------------------------------------------------------------------------

---

## Description

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

## Usage

```
visualize(dataset, xvar, yvar = "", type = "hist", facet_row = ".",
  facet_col = ".", color = "none", fill = "none", bins = 10,
  smooth = 1, 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")
type	Type of plot to create. One of Histogram ('hist'), 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	Group bar, histogram, 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 (not accessible in Radiant)
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://vnijs.github.io/radiant/base/visualize.html> for an example in Radiant

## Value

Generated plots

## Examples

```
visualize("diamonds", "carat", "price", type = "scatter", check = "loess")
visualize("diamonds", "price:x", type = "hist")
visualize("diamonds", "carat:x", yvar = "price", type = "scatter")
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")
```



---

win_launcher	Create a launcher and updater for Windows (.bat)
--------------	--------------------------------------------------

---

**Description**

Create a launcher and updater for Windows (.bat)

**Usage**

```
win_launcher(app = c("analytics", "marketing", "quant", "base"))
```

**Arguments**

app	App to run when the desktop icon is double-clicked ("analytics", "marketing", "quant", or "base"). Default is "analytics"
-----	---------------------------------------------------------------------------------------------------------------------------

**Details**

On Windows a file named 'radiant.bat' and one named 'update\_radiant.bat' will be put on the desktop. Double-click the file to launch the specified Radiant app or update Radiant to the latest version

**Examples**

```
if (interactive()) {  
  if (Sys.info()["sysname"] == "Windows") {  
    win_launcher()  
    fn <- paste0(Sys.getenv("USERPROFILE"), "/Desktop/radiant.bat")  
    if (!file.exists(fn))  
      stop("Windows launcher not created")  
    else  
      unlink(fn)  
  }  
}
```

# Index

## \*Topic **datasets**

- avengers, [4](#)
- city, [6](#)
- computer, [10](#)
- diamonds, [15](#)
- mp3, [34](#)
- newspaper, [35](#)
- publishers, [59](#)
- rndnames, [61](#)
- shopping, [66](#)
- superheroes, [91](#)
- titanic, [93](#)
- titanic\_pred, [93](#)
- toothpaste, [94](#)

avengers, [4](#)

changedata, [5](#)  
city, [6](#)  
clean\_loadings, [6](#)  
combinedata, [7](#)  
compare\_means, [8](#), [39](#), [75](#)  
compare\_props, [9](#), [39](#), [76](#)  
computer, [10](#)  
conjoint, [10](#), [40](#), [76](#), [77](#), [92](#), [93](#)  
conjoint\_profiles, [11](#), [18](#), [77](#)  
copy\_all, [12](#)  
copy\_from, [12](#), [71–73](#)  
correlation, [13](#), [41](#), [78](#)  
cross\_tabs, [14](#), [41](#), [42](#), [79](#)  
cv, [15](#)

diamonds, [15](#)  
dtree, [16](#), [42](#), [79](#)

explore, [16](#), [19](#), [30](#), [80](#)

factorizer, [17](#)  
ff\_design, [18](#)  
filterdata, [18](#)  
flip, [19](#)  
full\_factor, [19](#), [43](#), [63](#), [81](#)

getclass, [20](#)  
getdata, [21](#)

getsummary, [22](#)  
glm\_reg, [22](#), [44](#), [45](#), [56](#), [74](#), [81](#), [82](#)

hier\_clus, [23](#), [46](#), [82](#)

is\_empty, [24](#)  
is\_string, [25](#)  
iterms, [25](#)

kmeans\_clus, [26](#), [46](#), [47](#), [64](#), [83](#)  
kurtosi, [27](#)

launcher, [27](#)  
lin\_launcher, [27](#), [28](#)  
loadcsv, [28](#)

mac\_launcher, [27](#), [29](#)  
make\_dt, [30](#)  
make\_expl, [19](#), [30](#)  
make\_funs, [31](#)  
max\_rm, [31](#)  
mds, [32](#), [47](#), [84](#)  
mean\_rm, [33](#)  
median\_rm, [33](#)  
min\_rm, [34](#)  
mp3, [34](#)

newspaper, [35](#)  
nmissing, [35](#)

p05, [36](#)  
p25, [36](#)  
p75, [37](#)  
p95, [37](#)  
pivotr, [30](#), [31](#), [38](#), [48](#), [84](#)  
plot.compare\_means, [8](#), [38](#), [75](#)  
plot.compare\_props, [9](#), [39](#), [76](#)  
plot.conjoint, [11](#), [40](#), [77](#), [93](#)  
plot.correlation\_, [13](#), [40](#), [78](#)  
plot.cross\_tabs, [14](#), [41](#), [79](#)  
plot.dtree, [16](#), [42](#), [79](#)  
plot.full\_factor, [20](#), [43](#), [43](#), [81](#)  
plot.glm\_predict, [23](#), [43](#), [45](#), [56](#), [82](#)  
plot.glm\_reg, [23](#), [44](#), [44](#), [45](#), [56](#), [82](#)  
plot.hier\_clus, [24](#), [45](#), [46](#), [82](#)

plot.kmeans\_clus, 26, 46, 64, 83  
 plot.mds, 32, 47, 84  
 plot.pivotr, 48  
 plot.pmap, 48, 55, 85  
 plot.pre\_factor, 49, 58, 86  
 plot.reg\_predict, 51  
 plot.regression, 50, 52, 57, 60, 87  
 plot.repeater, 52  
 plot.simulator, 53, 68, 89  
 plot.single\_mean, 53, 68, 90  
 plot.single\_prop, 54, 69, 91  
 pmap, 49, 55, 85  
 pre\_factor, 49, 50, 58, 86  
 predict.glm\_reg, 23, 44, 45, 56, 74, 82  
 predict.regression, 51, 52, 57, 60, 74, 87  
 print.gtable, 58  
 publishers, 59  
  
 radiant, 59  
 radiant-package (radiant), 59  
 regression, 50–52, 57, 60, 74, 87  
 repeater, 52, 61, 87  
 rndnames, 61  
  
 sample\_size, 62, 88  
 sampling, 63, 88, 89  
 save\_factors, 63  
 save\_membership, 26, 47, 64, 83  
 sd\_rm, 65  
 serr, 65  
 set\_class, 66  
 shopping, 66  
 sig\_stars, 66  
 simulator, 53, 67, 79, 89  
 single\_mean, 53, 54, 68, 90  
 single\_prop, 54, 55, 69, 90, 91  
 skew, 70  
 sshh, 70  
 sshhr, 70  
 state\_init, 71, 72, 73  
 state\_multiple, 71, 72, 73  
 state\_single, 71, 72, 73  
 store\_glm, 74  
 store\_reg, 74  
 sum\_rm, 91  
 summary.compare\_means, 8, 39, 75  
 summary.compare\_props, 9, 39, 76  
 summary.conjoint, 11, 40, 76, 93  
 summary.conjoint\_profiles, 11, 18, 77  
 summary.correlation\_, 13, 41, 78  
 summary.cross\_tabs, 14, 42, 78  
 summary.dtree, 16, 42, 79  
 summary.explore, 17, 80  
 summary.full\_factor, 20, 80  
 summary.glm\_reg, 23, 44, 56, 81  
 summary.hier\_clus, 24, 46, 82, 82  
 summary.kmeans\_clus, 26, 47, 64, 83  
 summary.mds, 32, 47, 83  
 summary.pivotr, 30, 31, 48, 84  
 summary.pmap, 49, 55, 85  
 summary.pre\_factor, 50, 58, 86  
 summary.regression, 51, 52, 57, 60, 86  
 summary.repeater, 87  
 summary.sample\_size, 62, 88  
 summary.sampling, 63, 88  
 summary.simulator, 68, 89  
 summary.single\_mean, 53, 54, 68, 90  
 summary.single\_prop, 55, 69, 90  
 superheroes, 91  
  
 test\_specs, 92  
 the\_table, 92  
 titanic, 93  
 titanic\_pred, 93  
 toothpaste, 94  
  
 var\_check, 94  
 viewdata, 95  
 visualize, 95  
  
 win\_launcher, 27, 97