

# Package ‘radiant’

May 9, 2015

**Title** Business Analytics using R and Shiny

**Version** 0.2.10

**Date** 2015-5-8

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

**Depends** R (>= 3.1.0),  
magrittr (>= 1.5),  
ggplot2 (>= 1.0.0),  
tidyr (>= 0.2.0),  
dplyr (>= 0.4.1)

**Imports** car (>= 2.0.22),  
MASS (>= 7.3),  
gridExtra (>= 0.9.1),  
AlgDesign (>= 1.1.7.3),  
GPARotation (>= 2014.11.1),  
psych (>= 1.4.8.11),  
wordcloud (>= 2.5),  
markdown (>= 0.7.4),  
knitr (>= 1.8),  
ggdendro (>= 0.1.15),  
broom (>= 0.3.6),  
pryr (>= 0.1),  
yaml (>= 2.1.13),  
htmlwidgets (>= 0.3.3),  
rpivotTable (>= 0.1.3.4),  
shiny (>= 0.11.1.9004),  
shinyAce (>= 0.2.1),  
lubridate (>= 1.3.3),  
DT (>= 0.0.38)

**Suggests** rmarkdown (>= 0.4.2),  
ggvis (>= 0.4),  
devtools (>= 1.7.0),  
testthat (>= 0.9.1),  
covr (>= 0.2.0.9002)

**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
ca_the_table . . . . .	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 . . . . .	14
diamonds . . . . .	15
explore . . . . .	15
ff_design . . . . .	16
full_factor . . . . .	17
getclass . . . . .	18
getdata . . . . .	18
getsummary . . . . .	19
glm_reg . . . . .	19
hier_clus . . . . .	20
is_empty . . . . .	21
is_string . . . . .	22
kmeans_clus . . . . .	22
kurtosi . . . . .	23
launcher . . . . .	24
lin_launcher . . . . .	24
mac_launcher . . . . .	25
max_rm . . . . .	26
mds . . . . .	26
mean_rm . . . . .	27
median_rm . . . . .	28
min_rm . . . . .	28
mp3 . . . . .	29
newspaper . . . . .	29
nmissing . . . . .	30
p25 . . . . .	30
p75 . . . . .	31
plot.compare_means . . . . .	31
plot.compare_props . . . . .	32
plot.conjoint . . . . .	33
plot.correlation . . . . .	33
plot.cross_tabs . . . . .	34
plot.explore . . . . .	35

plot.full_factor . . . . .	36
plot.glm_predict . . . . .	36
plot.glm_reg . . . . .	37
plot.hier_clus . . . . .	38
plot.kmeans_clus . . . . .	39
plot.mds . . . . .	40
plot.pmap . . . . .	41
plot.pre_factor . . . . .	42
plot.regression . . . . .	42
plot.reg_predict . . . . .	44
plot.single_mean . . . . .	45
plot.single_prop . . . . .	45
pmap . . . . .	46
predict.glm_reg . . . . .	47
predict.regression . . . . .	48
pre_factor . . . . .	49
print.arrange . . . . .	49
publishers . . . . .	50
radiant . . . . .	50
regression . . . . .	51
rndnames . . . . .	52
sample_size . . . . .	52
sampling . . . . .	53
save_factors . . . . .	54
save_glm_resid . . . . .	54
save_membership . . . . .	55
save_reg_resid . . . . .	56
sd_rm . . . . .	56
serr . . . . .	57
set_class . . . . .	57
shopping . . . . .	58
sig_stars . . . . .	58
single_mean . . . . .	59
single_prop . . . . .	60
skew . . . . .	60
sshh . . . . .	61
sshhr . . . . .	61
state_init . . . . .	62
state_multiple . . . . .	63
state_single . . . . .	64
summary.compare_means . . . . .	65
summary.compare_props . . . . .	65
summary.conjoint . . . . .	66
summary.conjoint_profiles . . . . .	67
summary.correlation . . . . .	67
summary.cross_tabs . . . . .	68
summary.explore . . . . .	69
summary.full_factor . . . . .	70
summary.glm_reg . . . . .	70
summary.hier_clus . . . . .	71
summary.kmeans_clus . . . . .	72
summary.mds . . . . .	73

summary.pmap . . . . .	73
summary.pre_factor . . . . .	74
summary.regression . . . . .	75
summary.sample_size . . . . .	76
summary.sampling . . . . .	76
summary.single_mean . . . . .	77
summary.single_prop . . . . .	78
superheroes . . . . .	78
test_specs . . . . .	79
titanic . . . . .	79
titanic_pred . . . . .	80
toothpaste . . . . .	80
var_check . . . . .	81
visualize . . . . .	81
win_launcher . . . . .	82

<b>Index</b>	<b>84</b>
--------------	-----------

---

avengers	<i>Avengers</i>
----------	-----------------

---

**Description**

Avengers

**Usage**

data(avengers)

**Format**

A data frame with 7 rows and 4 variables

**Details**

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

---

ca_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**

ca\_the\_table(model, dat, ca\_indep\_var)

**Arguments**

model	Tidied model results (broom) output from <a href="#">conjoint</a> passed on by summary.conjoint
dat	Conjoint data
ca_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", ca_dep_var = "Rating", ca_indep_var = "Memory:Shape")
ca_the_table(result$model, result$dat, result$ca_indep_var)
```

---

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(ff_loadings, ff_cutoff = 0, ff_sort = FALSE, ff_round = 8)
```

**Arguments**

<code>ff_loadings</code>	Data.frame with loadings
<code>ff_cutoff</code>	Show only loadings with (absolute) values above <code>ff_cutoff</code> (default = 0)
<code>ff_sort</code>	Sort factor loadings
<code>ff_round</code>	Number of digits 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$ff_loadings, 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, dataset2, cmb_vars = "", cmb_type = "bind_rows",
  cmb_name = "cmb_data")
```

**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
dataset2	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
cmb_vars	Variables used to combine 'dataset' and 'dataset2'
cmb_type	The main bind and join types from the dplyr package are provided. 'inner_join' returns all rows from x with matching values in y, and all columns from x and y. If there are multiple matches between x and y, all match combinations are returned. 'left_join' returns all rows from x, and all columns from x and y. If there are multiple matches between x and y, all match combinations are returned. 'right_join' is equivalent to a 'left_join' for datasets y and x. 'full_join' combines two datasets, keeping rows and columns that appear in either. 'semi_join' returns all rows from x with matching values in y, keeping just columns from x. A semi join differs from an inner join because an inner join will return one row of x for each matching row of y, whereas a semi join will never duplicate rows of x. 'anti_join' returns all rows from x without matching values in y, keeping only columns from x.
cmb_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 'cmb\_name'. Else the combined dataset will be returned as 'cmb\_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, cmb_type = "bind_cols")
combinedata("avengers", "superheroes", cmb_type = "bind_cols")
avengers %>% combinedata(superheroes, cmb_type = "bind_rows")
```

compare\_means

*Compare means for two or more variables***Description**

Compare means for two or more variables

**Usage**

```
compare_means(dataset, cm_var1, cm_var2, data_filter = "",
  cm_paired = "independent", cm_alternative = "two.sided",
  cm_sig_level = 0.95, cm_adjust = "none")
```

**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
cm_var1	A numeric variable or factor selected for comparison
cm_var2	One or more numeric variables for comparison. If <code>cm_var1</code> is a factor only one variable can be selected and the mean of this variable is compared across (factor) levels of <code>cm_var1</code>
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")
cm_paired	Are samples indepent ("independent") or not ("paired")
cm_alternative	The alternative hypothesis ("two.sided", "greater" or "less")
cm_sig_level	Span of the confidence interval
cm_adjust	Adjustment for multiple comparisons ("none" or "bonf" for Bonferroni)

**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, cp_var1, cp_var2, data_filter = "", cp_levels = "",  
  cp_alternative = "two.sided", cp_sig_level = 0.95, cp_adjust = "none")
```

## 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>cp_var1</code>	A grouping variable to split the data for comparisons
<code>cp_var2</code>	The variable to calculate proportions for
<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")
<code>cp_levels</code>	The factor level selected for the proportion comparison
<code>cp_alternative</code>	The alternative hypothesis ("two.sided", "greater" or "less")
<code>cp_sig_level</code>	Span of the confidence interval
<code>cp_adjust</code>	Adjustment for multiple comparisons ("none" or "bonf" for Bonferroni)

## 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, ca_dep_var, ca_indep_var, data_filter = "",
  ca_rev = 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
ca_dep_var	The dependent variable (e.g., profile ratings)
ca_indep_var	Independent variables in the regression
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")
ca_rev	Reverse the values of the dependent variable ('ca_dep_var')

**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", ca_dep_var = "Rating", ca_indep_var = "Memory:Shape")
result <- mp3 %>% conjoint(ca_dep_var = "Rating", ca_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**

```
ca_prof <- readLines(system.file("examples/profiles-movie.txt", package='radiant'))
result <- conjoint_profiles("ca_prof")
rm(ca_prof, 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, cor_var, data_filter = "", cor_type = "pearson")
```

## 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
cor_var	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")
cor_type	Type of correlations to calculate. Options are "pearson", "spearman", and "kendall". "pearson" is the default

## 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",c("price:table"))
```

---

cross\_tabs

*Evaluate associations between categorical variables*


---

### Description

Evaluate associations between categorical variables

### Usage

```
cross_tabs(dataset, ct_var1, ct_var2, 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
ct_var1	A categorical variable
ct_var2	Another categorical variable
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/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")
```

---

cv

*Coefficient of variation*


---

### Description

Coefficient of variation

### Usage

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

**Arguments**

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

**Value**

Coefficient of variation

**Examples**

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

---

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")`

---

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

---

**Description**

Explore data

**Usage**

```
explore(dataset, expl_vars = "", data_filter = "", expl_byvar = "",  
        expl_fun = c("length", "mean_rm"))
```

**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
expl_vars	(Numerical) variables to summaries
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")
expl_byvar	Variable(s) to group data by before summarizing
expl_fun	Functions to use for summarizing

**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

[plot.explore](#) to plot summaries

**Examples**

```
result <- explore("diamonds", "price:x")
summary(result)
result <- explore("diamonds", "price", expl_byvar = "cut", expl_fun = c("length", "skew"))
summary(result)
```

---

ff\_design

---

*Function to generate a fractional factorial design*


---

**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

---

full\_factor

*Factor analysis (PCA)*


---

**Description**

Factor analysis (PCA)

**Usage**

```
full_factor(dataset, ff_var, data_filter = "", ff_meth = "PCA",
  ff_number = 2, ff_rotation = "varimax")
```

**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
ff_var	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")
ff_meth	Factor extraction method to use
ff_number	Number of factors to extract
ff_rotation	Apply varimax rotation or no rotation ("varimax" or "none")

**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"), ff_meth = "maxlik")
summary(result)
```

---

getclass	<i>Get variable class</i>
----------	---------------------------

---

**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 = "", na.rm = TRUE, filt = "", slice = "")
```

**Arguments**

dataset	Name of the dataframe
vars	Variables to extract from the dataframe
na.rm	Remove rows with missing values (default is TRUE)
filt	Filter to apply to the specified dataset. For example "price > 10000" if dataset is "diamonds" (default is "")
slice	Select a slice of the specified dataset. For example "1:10" for the first 10 rows or "n()-10:n()" for the last 10 rows (default is ""). Not in Radiant GUI

**Value**

Data.frame with specified columns and rows

**Examples**

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

---

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

---

**Description**

Create data.frame summary

**Usage**

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

**Arguments**

dat	Data.frame
dc	Class for each variable

**Details**

Used by Explore and Transform

---

glm_reg	<i>Generalized linear models (GLM)</i>
---------	--

---

**Description**

Generalized linear models (GLM)

**Usage**

```
glm_reg(dataset, glm_dep_var, glm_indep_var, data_filter = "",
  glm_levels = "", glm_link = "logit", glm_int_var = "", glm_check = "")
```

**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
glm_dep_var	The dependent variable in the logit (probit) model
glm_indep_var	Independent variables in the model
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")
glm_levels	The level in the dependent variable defined as <code>_success_</code>
glm_link	Link function for <code>_glm_</code> ('logit' or 'probit'). 'logit' is the default
glm_int_var	Interaction term to include in the model (not implement)
glm_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

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

---

hier\_clus

*Hierarchical cluster analysis*

---

**Description**

Hierarchical cluster analysis

**Usage**

```
hier_clus(dataset, hc_vars, data_filter = "", hc_dist = "sq.euclidian",
  hc_meth = "ward.D")
```

**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
hc_vars	Vector of 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")
hc_dist	Distance
hc_meth	Method

**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", hc_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)
```

---

kmeans_clus	<i>K-means cluster analysis</i>
-------------	---------------------------------

---

**Description**

K-means cluster analysis

**Usage**

```
kmeans_clus(dataset, km_vars, data_filter = "", km_hc_init = TRUE,  
  km_dist = "sq.euclidian", km_meth = "ward.D", km_seed = 1234,  
  km_nr_clus = 2)
```

**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
km_vars	Vector of 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")
km_hc_init	Use centers from hier_clus as the starting point
km_dist	Distance for hier_clus
km_meth	Method for hier_clus
km_seed	Random see to use for kmeans if km_hc_init is FALSE
km_nr_clus	Number of clusters to extract

**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

---

*Exporting the kurtosi function from the psych package*


---

**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("marketing", "quant", "base"))
```

### Arguments

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

### 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 for Linux (.sh)</i>
--------------	--

---

### Description

Create a launcher for Linux (.sh)

### Usage

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

### Arguments

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

### Details

On Linux a file named 'radiant.sh' will be put on the desktop. Double-click the file to launch the specified Radiant app



**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)
  }
}

```

mac\_launcher

*Create a launcher for Mac (.command)***Description**

Create a launcher for Mac (.command)

**Usage**

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

**Arguments**

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

**Details**

On Mac a file named 'radiant.command' will be put on the desktop. Double-click the file to launch the specified Radiant app

**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)
  }
}

```

---

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, mds_id1, mds_id2, mds_dis, data_filter = "",
      mds_method = "metric", mds_dim_number = 2)
```

**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
mds_id1	A character variable or factor with unique entries
mds_id2	A character variable or factor with unique entries
mds_dis	A numeric measure of brand dissimilarity
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> )
mds_method	Apply metric or non-metric MDS
mds_dim_number	Number of dimensions

**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))
```

---

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

---

**Description**

Min with na.rm = TRUE

**Usage**

```
min_rm(x)
```

**Arguments**

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

**Value**

Minimum value

**Examples**

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

---

mp3	<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))
```

---

p25	<i>25th percentile</i>
-----	------------------------

---

**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	<i>75th percentile</i>
-----	------------------------

---

**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))
```

---

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

---

**Description**

Plot method for the compare\_means function

**Usage**

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

**Arguments**

x	Return value from <a href="#">compare_means</a>
cm_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, cm_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, cp_plots = "props", shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">compare_props</a>
cp_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, cp_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, ca_plots = "pw", ca_scale_plot = FALSE,
     shiny = FALSE, ...)
```

### Arguments

x	Return value from <a href="#">conjoint</a>
ca_plots	Show either the part-worth ("pw") or importance-weights ("iw") plot
ca_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", ca_dep_var = "Rating", ca_indep_var = "Memory:Shape")
plot(result, ca_scale_plot = TRUE)
plot(result, ca_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

---

*Plot method for the cross\_tabs function*


---

**Description**

Plot method for the cross\_tabs function

**Usage**

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

**Arguments**

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

ct\_check              Show plots for variables ct\_var1 and ct\_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, ct_check = c("observed", "expected", "chi_sq"))
newspaper %>% cross_tabs("Income", "Newspaper") %>% plot(c("observed", "expected"))
```

---

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

---

**Description**

Plot method for the explore function

**Usage**

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

**Arguments**

x	Return value from <a href="#">explore</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/base/explore.html> for an example in Radiant. A plot will only be generated when a 'by' variable has been specified

**See Also**

[explore](#) to generate summaries

[summary.explore](#) to show summaries

**Examples**

```
result <- explore("diamonds", "price", expl_byvar = "cut", expl_fun = c("length", "skew"))
plot(result)
```

---

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", "HighEnd: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, glm_xvar = "", glm_facet_row = ".",
      glm_facet_col = ".", glm_color = "none", glm_conf_level = 0.95, ...)
```

**Arguments**

x	Return value from <code>predict.glm_reg</code> .
glm_xvar	Variable to display along the X-axis of the plot
glm_facet_row	Create vertically arranged subplots for each level of the selected factor variable
glm_facet_col	Create horizontally arranged subplots for each level of the selected factor variable
glm_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
glm_conf_level	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"), glm_levels = "Yes")
pred <- predict(result, glm_predict_cmd = "pclass = levels(pclass)")
plot(pred, glm_xvar = "pclass")
pred <- predict(result, glm_predict_cmd = "age = 0:100")
plot(pred, glm_xvar = "age")
pred <- predict(result, glm_predict_cmd = "pclass = levels(pclass), sex = levels(sex)")
plot(pred, glm_xvar = "pclass", glm_color = "sex")
pred <- predict(result, glm_predict_cmd = "pclass = levels(pclass), age = seq(0,100,20)")
plot(pred, glm_xvar = "pclass", glm_color = "age")
plot(pred, glm_xvar = "age", glm_color = "pclass")
pred <- predict(result, glm_predict_cmd="pclass=levels(pclass), sex=levels(sex), age=seq(0,100,20)")
plot(pred, glm_xvar = "age", glm_color = "sex", glm_facet_col = "pclass")
plot(pred, glm_xvar = "age", glm_color = "pclass", glm_facet_col = "sex")
pred <- predict(result, glm_predict_cmd="pclass=levels(pclass), sex=levels(sex), age=seq(0,100,5)")
plot(pred, glm_xvar = "age", glm_color = "sex", glm_facet_col = "pclass")
plot(pred, glm_xvar = "age", glm_color = "pclass", glm_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, glm_plots = "", glm_conf_level = 0.95,
     glm_coef_int = FALSE, shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">glm_reg</a>
glm_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
glm_conf_level	Confidence level to use for coefficient and odds confidence intervals (.95 is the default)
glm_coef_int	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"), glm_levels = "Yes")
plot(result, glm_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, hc_plots = c("scree", "diff"), hc_cutoff = 0.02,
     shiny = TRUE, ...)
```

**Arguments**

x	Return value from <a href="#">hier_clus</a>
hc_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
hc_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", hc_vars = c("v1:v6"))
plot(result, hc_plots = c("diff", "scree"), hc_cutoff = .05)
plot(result, hc_plots = "dendro", hc_cutoff = 0)
shopping %>% hier_clus(hc_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", km_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, mds_rev_dim = "", mds_fontsz = 1.3, ...)
```

**Arguments**

x	Return value from <a href="#">mds</a>
mds_rev_dim	Flip the axes in plots
mds_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, mds_rev_dim = 1:2)  
plot(result, mds_rev_dim = 1:2, mds_fontsz = 2)
```



---

`plot.pmap`*Plot method for the pmap function*

---

## Description

Plot method for the pmap function

## Usage

```
## S3 method for class 'pmap'  
plot(x, pmap_plot = "", pmap_scaling = 2.1,  
      pmap_fontsz = 1.3, ...)
```

## Arguments

<code>x</code>	Return value from <a href="#">pmap</a>
<code>pmap_plot</code>	Components to include in the plot ("brand", "attr"). If data on preferences is available use "pref" to add preference arrows to the plot
<code>pmap_scaling</code>	Arrow scaling in the brand map
<code>pmap_fontsz</code>	Font size to use in plots
<code>...</code>	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", "HighEnd:Business")  
plot(result, pmap_plot = "brand")  
plot(result, pmap_plot = c("brand", "attr"))  
plot(result, pmap_plot = c("brand", "attr"))  
plot(result, pmap_scaling = 1, pmap_plot = c("brand", "attr"))  
result <- pmap("computer", "Brand", "HighEnd:Dated",  
              pmap_pref = c("Innovative", "Business"))  
plot(result, pmap_plot = 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, reg_plots = "", reg_lines = "",  
      reg_conf_level = 0.95, reg_coef_int = FALSE, shiny = FALSE, ...)
```

**Arguments**

x	Return value from <a href="#">regression</a>
reg_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
reg_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")
reg_conf_level	Confidence level used to estimate confidence intervals (.95 is the default)
reg_coef_int	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, reg_plots = "dashboard")
plot(result, reg_plots = "dashboard", reg_lines = c("line","loess"))
plot(result, reg_plots = "coef", reg_coef_int = TRUE)
plot(result, reg_plots = "coef", reg_conf_level = .99, reg_coef_int = TRUE)
plot(result, reg_plots = "hist")
plot(result, reg_plots = "scatter", reg_lines = c("line","loess"))
plot(result, reg_plots = "correlations")
plot(result, reg_plots = "leverage")
plot(result, reg_plots = "resid_pred", reg_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, reg_xvar = "", reg_facet_row = ".",
     reg_facet_col = ".", reg_color = "none", reg_conf_level = 0.95, ...)
```

## Arguments

x	Return value from <a href="#">predict.regression</a> .
reg_xvar	Variable to display along the X-axis of the plot
reg_facet_row	Create vertically arranged subplots for each level of the selected factor variable
reg_facet_col	Create horizontally arranged subplots for each level of the selected factor variable
reg_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
reg_conf_level	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, reg_predict_cmd = "carat = 1:10")
plot(pred, reg_xvar = "carat")
result <- regression("diamonds", "price", c("carat","clarity"), reg_int_var = "carat:clarity")
dpred <- getdata("diamonds") %>% slice(1:100)
pred <- predict(result, reg_predict_data = "dpred")
plot(pred, reg_xvar = "carat", reg_color = "clarity")
rm(dpred, envir = .GlobalEnv)
```

---

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, sm_plots = "hist", shiny = FALSE, ...)
```

### Arguments

x	Return value from <a href="#">single_mean</a>
sm_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 (sm_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", sm_comp_value = 3500)  
plot(result, sm_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, sp_plots = "hist", shiny = FALSE, ...)
```

**Arguments**

x	Return value from <code>single_prop</code>
sp_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 (sp_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", sp_levels = "IF", sp_comp_value = 0.05)
plot(result, sp_plots = c("hist", "simulate"))
result <- single_prop("titanic","pclass", sp_levels = "1st")
plot(result, sp_plots = c("hist","simulate"))
```

---

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

---

**Description**

Attribute based brand maps

**Usage**

```
pmap(dataset, pmap_brand, pmap_attr, data_filter = "", pmap_pref = "",
      pmap_dim_number = 2)
```

**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
pmap_brand	A character variable with brand names
pmap_attr	Names of numeric 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")
pmap_pref	Names of numeric brand preference measures
pmap_dim_number	Number of dimensions

**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", "HighEnd:Business")
```

---

predict.glm\_reg

*Predict method for the glm\_reg function*

---

**Description**

Predict method for the glm\_reg function

**Usage**

```
## S3 method for class 'glm_reg'
predict(object, glm_predict_cmd = "",
        glm_predict_data = "", ...)
```

**Arguments**

object	Return value from <a href="#">glm_reg</a>
glm_predict_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)')
glm_predict_data	Provide the name of a dataframe to generate predictions (e.g., "titanic"). The dataset must contain all columns used in the estimation
...	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"), glm_levels = "Yes")
predict(result, glm_predict_cmd = "pclass = levels(pclass)")
glm_reg("titanic", "survived", c("pclass","sex"), glm_levels = "Yes") %>%
  predict(glm_predict_cmd = "sex = c('male','female')")
```

---

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, reg_predict_cmd = "",
        reg_predict_data = "", reg_conf_level = 0.95, ...)
```

**Arguments**

object	Return value from <a href="#">regression</a>
reg_predict_cmd	Command used to generate data for prediction
reg_predict_data	Name of the dataset to use for prediction
reg_conf_level	Confidence level used to estimate confidence intervals (.95 is the default)
...	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, reg_predict_cmd = "carat = 1:10")
predict(result, reg_predict_cmd = "clarity = levels(clarity)")
result <- regression("diamonds", "price", c("carat","clarity"), reg_int_var = c("carat:clarity"))
dpred <- getdata("diamonds") %>% slice(1:10)
predict(result, reg_predict_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, pf_var, 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
pf_var	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.arrange	<i>Exporting the print.arrange method from the gridExtra package</i>
---------------	--

---

**Description**

Exporting the print.arrange method from the gridExtra package

**Details**

Used to print plots generated using arrangeGrob

---

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("marketing", "quant", "base"))
```

**Arguments**

app	Choose the app to run. Either "base", "quant", or "marketing". "marketing" is the default
-----	---

**Details**

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

**Examples**

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

---

regression

---

*Linear regression using OLS*

---

**Description**

Linear regression using OLS

**Usage**

```
regression(dataset, reg_dep_var, reg_indep_var, data_filter = "",  
           reg_int_var = "", reg_check = "")
```

**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
reg_dep_var	The dependent variable in the regression
reg_indep_var	Independent variables in the regression
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")
reg_int_var	Interaction terms to include in the model
reg_check	"standardize" to see standardized coefficient estimates. "stepwise" to apply stepwise selection of variables in estimation

**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"), reg_check = "standardize")
```

---

<code>rndnames</code>	<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")`

---

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

---

**Description**

Sample size calculation

**Usage**

```
sample_size(ss_type = "mean", ss_mean_err = 2, ss_mean_s = 10,
  ss_prop_err = 0.1, ss_prop_p = 0.5, ss_z = 1.96, ss_incidence = 1,
  ss_response = 1, ss_pop_correction = "no", ss_pop_size = 1000000)
```

**Arguments**

<code>ss_type</code>	Choose "mean" or "proportion"
<code>ss_mean_err</code>	Acceptable Error for Mean
<code>ss_mean_s</code>	Standard deviation for Mean
<code>ss_prop_err</code>	Acceptable Error for Proportion
<code>ss_prop_p</code>	Initial proportion estimate for Proportion
<code>ss_z</code>	Z-value
<code>ss_incidence</code>	Incidence rate (i.e., fraction of valid respondents)
<code>ss_response</code>	Response rate
<code>ss_pop_correction</code>	Apply correction for population size ("yes","no")
<code>ss_pop_size</code>	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(ss_type = "mean", ss_mean_err = 2, ss_mean_s = 10)
```

---

sampling	<i>Simple random sampling</i>
----------	-------------------------------

---

**Description**

Simple random sampling

**Usage**

```
sampling(dataset, smp_var, smp_sample_size, data_filter = "",
  smp_print_full = TRUE)
```

**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>smp_var</code>	The variable to sample from
<code>smp_sample_size</code>	Number of units to select
<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")
<code>smp_print_full</code>	Print full sampling frame. Default is TRUE

**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_glm_resid	Save residuals generated in the glm_reg function
----------------	--

---

**Description**

Save residuals generated in the glm\_reg function

**Usage**

```
save_glm_resid(object)
```

**Arguments**

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

**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", glm_levels = "Yes")
save_glm_resid(result)
head(titanic)
```

---

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", km_vars = c("v1:v6"))
save_membership(result)
head(shopping)
```

---

save_reg_resid	<i>Save regression residuals</i>
----------------	----------------------------------

---

**Description**

Save regression residuals

**Usage**

```
save_reg_resid(object)
```

**Arguments**

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

**Details**

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

**Examples**

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

---

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

---

**Description**

Standard deviation with na.rm = TRUE

**Usage**

```
sd_rm(x)
```

**Arguments**

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

**Value**

Standard deviation

**Examples**

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



---

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

---

**Description**

Standard error

**Usage**

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

**Arguments**

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

**Value**

Standard error

**Examples**

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

---

set_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))
```

---

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, sm_var, data_filter = "", sm_comp_value = 0,  
            sm_alternative = "two.sided", sm_sig_level = 0.95)
```

## 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
sm_var	The variable selected for the mean comparison
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")
sm_comp_value	Population value to compare to the sample mean
sm_alternative	The alternative hypothesis ("two.sided", "greater", or "less")
sm_sig_level	Span for the confidence interval

## 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	<i>Compare a sample proportion to a population proportion</i>
-------------	---

---

### Description

Compare a sample proportion to a population proportion

### Usage

```
single_prop(dataset, sp_var, data_filter = "", sp_levels = "",
  sp_comp_value = 0.5, sp_alternative = "two.sided", sp_sig_level = 0.95)
```

### 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
sp_var	The variable selected for the proportion comparison
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")
sp_levels	The factor level selected for the proportion comparison
sp_comp_value	Population value to compare to the sample proportion
sp_alternative	The alternative hypothesis ("two.sided", "greater", or "less")
sp_sig_level	Span of the confidence interval

### 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", sp_levels = "IF", sp_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`*Hide warnings and messages and return invisible*

---

**Description**

Hide warnings and messages and return invisible

**Usage**

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

**Arguments**

...                      Inputs to keep quiet

**Details**

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

**Examples**

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

---

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

---

**Description**

Hide warnings and messages and return result

**Usage**

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

**Arguments**

...                      Inputs to keep quiet

**Details**

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

**Examples**

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

---

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"))
```



---

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, ca_vif = FALSE, ...)
```

**Arguments**

object	Return value from <a href="#">conjoint</a>
ca_vif	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", ca_dep_var = "Rating", ca_indep_var = "Memory:Shape")
summary(result, ca_vif = TRUE)
mp3 %>% conjoint(ca_dep_var = "Rating", ca_indep_var = "Memory:Shape") %>% summary(., ca_vif = 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

<code>object</code>	Return value from <a href="#">conjoint_profiles</a>
<code>...</code>	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

```
ca_prof <- readLines(system.file("examples/profiles-movie.txt", package='radiant'))  
result <- conjoint_profiles("ca_prof")  
summary(result)  
rm(ca_prof, 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, cor_cutoff = 0, ...)
```

**Arguments**

object	Return value from <a href="#">correlation</a>
cor_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, cor_cutoff = .3)
diamonds %>% correlation("price:clarity") %>% summary
```

---

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

---

**Description**

Summary method for the cross\_tabs function

**Usage**

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

**Arguments**

object	Return value from <a href="#">cross_tabs</a>
ct_check	Show table(s) for variables ct_var1 and ct_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, ct_check = c("observed", "expected", "chi_sq"))
newspaper %>% cross_tabs("Income", "Newspaper") %>% summary("observed")
```

---

summary.explore

*Summary method for the explore function*


---

**Description**

Summary method for the explore function

**Usage**

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

**Arguments**

object	Return value from <a href="#">explore</a>
...	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

[plot.explore](#) to plot summaries

**Examples**

```
result <- explore("diamonds", "price:x")
summary(result)
result <- explore("diamonds", "price", expl_byvar = "cut", expl_fun = c("length", "skew"))
summary(result)
diamonds %>% explore("price:x") %>% summary
diamonds %>% explore("price", expl_byvar = "cut", expl_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, ff_cutoff = 0, ff_sort = FALSE, ...)
```

## Arguments

object	Return value from <a href="#">full_factor</a>
ff_cutoff	Show only loadings with (absolute) values above ff_cutoff (default = 0)
ff_sort	Sort factor loadings
...	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","depth","table","x"))
summary(result)
summary(result, ff_cutoff = 0, ff_sort = FALSE)
summary(result, ff_cutoff = 0, ff_sort = TRUE)
summary(result, ff_cutoff = .5, ff_sort = TRUE)
diamonds %>% full_factor(c("price","carat","depth","table","x")) %>% summary
diamonds %>% full_factor(c("price","carat","depth","table","x")) %>% summary(ff_cutoff = .5)
```

---

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

---

## Description

Summary method for the glm\_reg function

**Usage**

```
## S3 method for class 'glm_reg'
summary(object, glm_sum_check = "", glm_conf_level = 0.95,
        glm_test_var = "", ...)
```

**Arguments**

object	Return value from <code>glm_reg</code>
glm_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.
glm_conf_level	Confidence level to use for coefficient and odds confidence intervals (.95 is the default)
glm_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", glm_levels = "Yes")
summary(result, glm_test_var = "pclass")
res <- glm_reg("titanic", "survived", c("pclass", "sex"), glm_int_var="pclass:sex", glm_levels="Yes")
summary(res, glm_sum_check = c("vif", "confint", "odds"))
titanic %>% glm_reg("survived", c("pclass", "sex", "age"), glm_levels = "Yes") %>% summary("vif")
```

---

summary.hier\_clus

*Summary method for the hier\_clus function*


---

**Description**

Summary method for the hier\_clus function

**Usage**

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

**Arguments**

object            Return value from [hier\\_clus](#)  
 ...              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", hc_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 [kmeans\\_clus](#)  
 ...              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", km_vars = c("v1:v6"))
summary(result)
shopping %>% kmeans_clus(km_vars = c("v1:v6"), km_nr_clus = 3) %>% summary
```



summary.mds

*Summary method for the mds function***Description**

Summary method for the mds function

**Usage**

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

**Arguments**

object	Return value from <a href="#">mds</a>
mds_round	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, mds_round = 2)
city %>% mds("from", "to", "distance") %>% summary
```

summary.pmap

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

Summary method for the pmap function

**Usage**

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

**Arguments**

object	Return value from <a href="#">pmap</a>
pmap_cutoff	Show only loadings with (absolute) values above pmap_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", "HighEnd:Business")
summary(result)
summary(result, pmap_cutoff = .3)
result <- pmap("computer", "Brand", "HighEnd:Dated", pmap_pref = c("Innovative", "Business"))
summary(result)
computer %>% pmap("Brand", "HighEnd:Dated", pmap_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", "HighEnd: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, reg_sum_check = "",
        reg_conf_level = 0.95, reg_test_var = "", ...)
```

**Arguments**

object	Return value from <a href="#">regression</a>
reg_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.
reg_conf_level	Confidence level used to estimate confidence intervals (.95 is the default)
reg_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, reg_sum_check = c("rmse", "sumsquares", "vif", "confint"), reg_test_var = "clarity")
result <- regression("shopping", "v1", c("v2", "v3"))
summary(result, reg_test_var = "v2")
shopping %>% regression("v1", "v2:v6") %>% summary
```

---

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(ss_type = "mean", ss_mean_err = 2, ss_mean_s = 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, ...)
```

**Arguments**

object	Return value from <a href="#">sampling</a>
...	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

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

---

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", sp_levels = "IF", sp_comp_value = 0.05)
summary(result)
diamonds %>% single_prop("clarity", sp_levels = "IF", sp_comp_value = 0.05) %>% summary
```

---

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 <code>_regression_</code> or <code>_glm_</code>
int_var	Interaction terms specified

**Details**

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

**Value**

'test\_var' is a vector of variables to test

**Examples**

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

---

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>
------------	-----------------------------

---

**Description**

Toothpaste attitudes

**Usage**

```
data(toothpaste)
```

**Format**

A data frame with 60 rows and 10 variables

**Details**

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(indep_var, cn, int_var = "")
```

### Arguments

indep_var	List of independent variables provided to _regression_ or _glm_
cn	Column names for all independent variables in _dat_
int_var	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, 'indep\_var' is the list of independent variables, and int\_var are interaction terms

### Examples

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

---

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, viz_xvar, viz_yvar = "none", data_filter = "",
  viz_type = "hist", viz_facet_row = ".", viz_facet_col = ".",
  viz_color = "none", viz_bins = 10, viz_smooth = 1, viz_check = "",
  viz_axes = "", 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
viz_xvar	One or more variables to display along the X-axis of the plot
viz_yvar	Variable to display along the Y-axis of the plot (default = "none")
data_filter	Expression used to filter the dataset. This should be a string (e.g., "price > 10000")
viz_type	Type of plot to create. One of Histogram ('hist'), Density ('density'), Scatter ('scatter'), Line ('line'), Bar ('bar'), or Box-plot ('box')
viz_facet_row	Create vertically arranged subplots for each level of the selected factor variable
viz_facet_col	Create horizontally arranged subplots for each level of the selected factor variable
viz_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
viz_bins	Number of bins used for a histogram (not accessible in Radiant)
viz_smooth	Adjust the flexibility of the loess line for scatter plots (not accessible in Radiant)
viz_check	Add a regression line ("line"), a loess line ("loess"), or jitter ("jitter") to a scatter plot
viz_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")
shiny	Did the function call originate inside a shiny app

**Details**

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

**Value**

Generated plots

**Examples**

```
visualize("diamonds", "carat", "price", viz_type = "scatter", viz_check = "loess")
visualize("diamonds", "price:x", viz_type = "hist")
visualize("diamonds", "carat:x", viz_yvar = "price", viz_type = "scatter")
diamonds %>% visualize(c("price", "carat", "depth"), viz_type = "density")
```

---

win\_launcher

*Create a launcher for Windows (.bat)*


---

**Description**

Create a launcher for Windows (.bat)

**Usage**

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

**Arguments**

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

**Details**

On Windows a file named 'radiant.bat' will be put on the desktop. Double-click the file to launch the specified Radiant app

**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, 29
- newspaper, 29
- publishers, 50
- rndnames, 52
- shopping, 58
- superheroes, 78
- titanic, 79
- titanic\_pred, 80
- toothpaste, 80

avengers, 4

ca\_the\_table, 4

changedata, 5

city, 6

clean\_loadings, 6

combinedata, 7

compare\_means, 8, 31, 32, 65

compare\_props, 9, 32, 65, 66

computer, 10

conjoint, 5, 10, 33, 66

conjoint\_profiles, 11, 17, 67

copy\_all, 12

copy\_from, 12, 62–64

correlation, 13, 34, 68

cross\_tabs, 14, 34, 35, 68, 69

cv, 14

diamonds, 15

explore, 15, 35, 69

ff\_design, 16

full\_factor, 17, 36, 54, 70

getclass, 18

getdata, 18

getsummary, 19

glm\_reg, 19, 37, 38, 47, 54, 71

hier\_clus, 20, 39, 72

is\_empty, 21

is\_string, 22

kmeans\_clus, 22, 39, 40, 55, 72

kurtosi, 23

launcher, 24

lin\_launcher, 24, 24

mac\_launcher, 24, 25

max\_rm, 26

mds, 26, 40, 73

mean\_rm, 27

median\_rm, 28

min\_rm, 28

mp3, 29

newspaper, 29

nmissing, 30

p25, 30

p75, 31

plot.compare\_means, 8, 31, 65

plot.compare\_props, 9, 32, 66

plot.conjoint, 5, 11, 33, 66

plot.correlation, 13, 33, 68

plot.cross\_tabs, 14, 34, 69

plot.explore, 16, 35, 69

plot.full\_factor, 17, 36, 36, 70

plot.glm\_predict, 20, 36, 38, 47, 71

plot.glm\_reg, 20, 37, 37, 38, 47, 71

plot.hier\_clus, 21, 38, 39, 72

plot.kmeans\_clus, 23, 39, 55, 72

plot.mds, 27, 40, 73

plot.pmap, 41, 47, 74

plot.pre\_factor, 42, 49, 74

plot.reg\_predict, 44

plot.regression, 42, 44, 48, 51, 75

plot.single\_mean, 45, 59, 77

plot.single\_prop, 45, 60, 78

pmap, 41, 46, 74

pre\_factor, 42, 49, 74

predict.glm\_reg, 20, 37, 38, 47, 71

`predict.regression`, [43](#), [44](#), [48](#), [51](#), [75](#)  
`print.arrange`, [49](#)  
`publishers`, [50](#)  
  
`radiant`, [50](#)  
`radiant-package (radiant)`, [50](#)  
`regression`, [43](#), [44](#), [48](#), [51](#), [56](#), [75](#)  
`rndnames`, [52](#)  
  
`sample_size`, [52](#), [76](#)  
`sampling`, [53](#), [76](#), [77](#)  
`save_factors`, [54](#)  
`save_glm_resid`, [54](#)  
`save_membership`, [23](#), [40](#), [55](#), [72](#)  
`save_reg_resid`, [56](#)  
`sd_rm`, [56](#)  
`serr`, [57](#)  
`set_class`, [57](#)  
`shopping`, [58](#)  
`sig_stars`, [58](#)  
`single_mean`, [45](#), [59](#), [77](#)  
`single_prop`, [46](#), [60](#), [78](#)  
`skew`, [60](#)  
`sshh`, [61](#)  
`sshhr`, [61](#)  
`state_init`, [62](#), [63](#), [64](#)  
`state_multiple`, [62](#), [63](#), [64](#)  
`state_single`, [62](#), [63](#), [64](#)  
`summary.compare_means`, [8](#), [32](#), [65](#)  
`summary.compare_props`, [9](#), [32](#), [65](#)  
`summary.conjoint`, [5](#), [11](#), [33](#), [66](#)  
`summary.conjoint_profiles`, [11](#), [17](#), [67](#)  
`summary.correlation`, [13](#), [34](#), [67](#)  
`summary.cross_tabs`, [14](#), [35](#), [68](#)  
`summary.explore`, [16](#), [35](#), [69](#)  
`summary.full_factor`, [17](#), [70](#)  
`summary.glm_reg`, [20](#), [37](#), [47](#), [70](#)  
`summary.hier_clus`, [21](#), [39](#), [71](#), [72](#)  
`summary.kmeans_clus`, [23](#), [40](#), [55](#), [72](#)  
`summary.mds`, [27](#), [40](#), [73](#)  
`summary.pmap`, [41](#), [47](#), [73](#)  
`summary.pre_factor`, [42](#), [49](#), [74](#)  
`summary.regression`, [43](#), [44](#), [48](#), [51](#), [75](#)  
`summary.sample_size`, [53](#), [76](#)  
`summary.sampling`, [53](#), [76](#)  
`summary.single_mean`, [45](#), [59](#), [77](#)  
`summary.single_prop`, [46](#), [60](#), [78](#)  
`superheroes`, [78](#)  
  
`test_specs`, [79](#)  
`titanic`, [79](#)  
`titanic_pred`, [80](#)  
`toothpaste`, [80](#)  
  
`var_check`, [81](#)  
`visualize`, [81](#)  
  
`win_launcher`, [24](#), [82](#)