

Package ‘radiant’

March 26, 2015

Title Business Analytics using R and Shiny

Version 0.1.85

Date 2015-3-26

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

Depends R (>= 3.1.0),
lubridate (>= 1.3.3),
ggplot2 (>= 1.0.0),
dplyr (>= 0.4.1),
magrittr (>= 1.5)

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),
tidyr (>= 0.2.0),
pryr (>= 0.1),
htmlwidgets (>= 0.3.2),
rpivotTable (>= 0.1.2.6),
shiny (>= 0.11.1),
shinyAce (>= 0.2.1)

Suggests rmarkdown (>= 0.4.2),
ggvis (>= 0.4),
testthat (>= 0.9.1)

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:

| | |
|------------------------------|----|
| ca_the_table | 3 |
| changedata | 4 |
| city | 5 |
| compare_means | 5 |
| compare_props | 6 |
| computer | 7 |
| conjoint | 7 |
| conjoint_profiles | 8 |
| correlation | 9 |
| cross_tabs | 10 |
| diamonds | 10 |
| ff_design | 11 |
| full_factor | 11 |
| getdata | 12 |
| glm_reg | 13 |
| hier_clus | 14 |
| kmeans_clus | 15 |
| kurtosi | 16 |
| mac_launcher | 16 |
| mds | 17 |
| mergedata | 18 |
| mp3 | 19 |
| newspaper | 19 |
| plot.compare_means | 20 |
| plot.compare_props | 20 |
| plot.conjoint | 21 |
| plot.correlation | 22 |
| plot.cross_tabs | 22 |
| plot.full_factor | 23 |
| plot.glm_predict | 24 |
| plot.glm_reg | 25 |
| plot.hier_clus | 26 |
| plot.kmeans_clus | 27 |
| plot.mds | 27 |
| plot.pmap | 28 |
| plot.pre_factor | 29 |
| plot.regression | 30 |
| plot.reg_predict | 31 |
| plot.single_mean | 32 |
| plot.single_prop | 32 |
| pmap | 33 |
| predict.glm_reg | 34 |
| predict.regression | 35 |
| pre_factor | 36 |
| radiant | 36 |
| regression | 37 |
| rndnames | 38 |
| sample_size | 38 |
| sampling | 39 |
| save_factors | 40 |

| | |
|-------------------------------------|-----------|
| save_glm_resid | 40 |
| save_membership | 41 |
| save_reg_resid | 42 |
| set_class | 42 |
| shopping | 43 |
| sig_stars | 43 |
| single_mean | 44 |
| single_prop | 45 |
| skew | 45 |
| sshh | 46 |
| sshhr | 46 |
| summary.compare_means | 47 |
| summary.compare_props | 47 |
| summary.conjoint | 48 |
| summary.conjoint_profiles | 49 |
| summary.correlation | 49 |
| summary.cross_tabs | 50 |
| summary.full_factor | 51 |
| summary.glm_reg | 51 |
| summary.hier_clus | 52 |
| summary.kmeans_clus | 53 |
| summary.mds | 54 |
| summary.pmap | 54 |
| summary.pre_factor | 55 |
| summary.regression | 56 |
| summary.sample_size | 57 |
| summary.sampling | 57 |
| summary.single_mean | 58 |
| summary.single_prop | 59 |
| test_check | 59 |
| titanic | 60 |
| titanic_pred | 60 |
| toothpaste | 61 |
| var_check | 61 |
| visualize | 62 |
| win_launcher | 63 |
| Index | 64 |

ca_the_table

*Function to calculate the PW and IW table for conjoint***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 conjoint 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

Change data

Description

Change data

Usage

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

Arguments

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

Value

None

Examples

```
## Not run:
r_data <- list()
r_data$dat <- data.frame(a = 1:20)
changedata("dat", 20:1, "b")
head(r_data$dat)

## End(Not run)
```

| | |
|------|-----------------------|
| 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")`

| | |
|---------------|--|
| compare_means | <i>Compare means for two or more variables</i> |
|---------------|--|

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., <code>Data > View</code> to filter the dataset in Radiant. The expression should be a string (e.g., <code>"price > 10000"</code>) |
| 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 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")
```

| | |
|----------------------------|--|
| <code>compare_props</code> | <i>Compare proportions across groups</i> |
|----------------------------|--|

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 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")
```

| | |
|----------|--|
| 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(dataset = "mp3", 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 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")
```

| | |
|-------------|---|
| 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 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")
```

diamonds

Diamond prices

Description

Diamond prices

Usage

```
data(diamonds)
```

Format

A data frame with 3000 rows and 10 variables

Details

A sample of 3,000 from the diamonds dataset bundled with ggplot2. Description provided in `attr(diamonds,"description")`

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 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 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)
```

getdata

Get data for analysis functions

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")
```

| | |
|---------|--|
| 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 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")
```

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

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 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"))
```

| | |
|-------------|---------------------------------|
| 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 for an example in Radiant

Value

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

See Also

[summary.kmeans_clus](#) to summarize results
[plot.kmeans_clus](#) to plot results
[save_membership](#) to add cluster membership to the selected dataset

Examples

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

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

Description

Exporting the kurtosi function from the psych package

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

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)
  }
}
```

| | |
|-----|---|
| 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 > View</code> to filter the dataset in Radiant. The expression should be a string (e.g., <code>"price > 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")
result <- mds("diamonds", "clarity", "cut", "price")
summary(result)
```

mergedata

*Merge datasets using dplyr's join functions***Description**

Merge datasets using dplyr's join functions

Usage

```
mergedata(dataset, dataset2, merge_vars = "", merge_type = "inner_join",
  merge_name = paste0("merged_", 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 |
| dataset2 | Dataset name (string) to merge with 'dataset'. This can be a dataframe in the global environment or an element in an <code>r_data</code> list from Radiant |
| merge_vars | Variables used to merge/join 'dataset' and 'dataset2' |
| merge_type | The main 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. '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 |
| merge_name | Name for the merged dataset |

Details

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

Value

If (reactive) list 'r_data' exists the merged dataset added as 'merge_name'. Else the merged dataset will be returned as 'merge_name'

Examples

```
mergedata("titanic", "titanic_pred", c("pclass", "sex", "age")) %>% head
```

| | |
|-----|--------------------------------------|
| 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")`

plot.compare_means *Plot method for the compare_means function*

Description

Plot method for the compare_means function

Usage

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

Arguments

| | |
|----------|---|
| x | Return value from compare_means |
| cm_plots | One or more plots ("bar", "box", or "density") |
| ... | further arguments passed to or from other methods |

Details

See 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 *Plot method for the compare_props function*

Description

Plot method for the compare_props function

Usage

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

Arguments

| | |
|----------|--|
| x | Return value from compare_props |
| cp_plots | One or more plots of proportions or counts ("props" or "counts") |
| ... | further arguments passed to or from other methods |

Details

See 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

Plot method for the conjoint function

Description

Plot method for the conjoint function

Usage

```
## S3 method for class 'conjoint'
plot(x, ca_plots = "pw", ca_scale_plot = FALSE, ...)
```

Arguments

| | |
|---------------|---|
| x | Return value from conjoint |
| 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 |
| ... | 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)
```

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

Description

Plot method for the cross_tabs function

Usage

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

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., $(o - e) / e$) |
| ... | further arguments passed to or from other methods |

Details

See 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"))
```

| | |
|------------------|---|
| 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, ...)
```

Arguments

| | |
|-----|---|
| x | Return value from full_factor |
| ... | further arguments passed to or from other methods |

Details

See 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

*Plot method for the predict.glm_reg function***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 predict.glm_reg . |
| 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 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, ...)
```

Arguments

| | |
|----------------|---|
| x | Return value from glm_reg |
| 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 |
| ... | further arguments passed to or from other methods |

Details

See 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 | <i>Plot method for the hier_clus function</i> |
|----------------|---|

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,
     ...)
```

Arguments

| | |
|-----------|---|
| x | Return value from hier_clus |
| 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 |
| ... | further arguments passed to or from other methods |

Details

See 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)
```

| | |
|------------------|------------------------------------|
| 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, ...)
```

Arguments

| | |
|-----|---|
| x | Return value from kmeans_clus |
| ... | further arguments passed to or from other methods |

Details

See 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 | <i>Plot method for the mds function</i> |
|----------|---|

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 mds |
| 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

| | |
|--------------|--|
| x | Return value from pmap |
| pmap_plot | Components to include in the plot ("brand", "attr"). If data on preferences is available use "pref" to add preference arrows to the plot |
| pmap_scaling | Arrow scaling in the brand map |
| pmap_fontsz | Font size to use in plots |
| ... | further arguments passed to or from other methods |

Details

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

See Also

[pmap](#) to calculate results

[summary.pmap](#) to plot results

Examples

```
result <- pmap("computer", "Brand", "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

Plot method for the pre_factor function

Description

Plot method for the pre_factor function

Usage

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

Arguments

| | |
|-----|---|
| x | Return value from pre_factor |
| ... | further arguments passed to or from other methods |

Details

See 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, ...)
```

Arguments

| | |
|----------------|--|
| x | Return value from regression |
| 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 |
| ... | 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 predict.regression . |
| 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")

```

| | |
|------------------|---|
| 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", ...)
```

Arguments

| | |
|----------|--|
| x | Return value from single_mean |
| 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 |
| ... | further arguments passed to or from other methods |

Details

See 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", ...)
```


Arguments

| | |
|----------|--|
| x | Return value from single_prop |
| 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 |
| ... | further arguments passed to or from other methods |

Details

See 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"))
```

| | |
|------|-----------------------------------|
| 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 | <i>Predict method for the glm_reg function</i> |
|-----------------|--|

Description

Predict method for the glm_reg function

Usage

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

Arguments

| | |
|------------------|--|
| object | Return value from glm_reg |
| 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 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)")
predict(result, 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, reg_save_pred = FALSE,
        ...)
```

Arguments

| | |
|------------------|---|
| object | Return value from regression |
| 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) |
| reg_save_pred | Save predicted values to a csv file |
| ... | 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")
```

| | |
|------------|---|
| 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 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"))
```

| | |
|---------|----------------|
| 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()
}
```

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

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 step-wise 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")
```

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

Description

100 random names

Usage

```
data(rndnames)
```

Format

A data frame with 100 rows and 2 variables

Details

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

| | |
|-------------|--------------------------------|
| sample_size | <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

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

Details

See 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 full_factor |
|--------|---|

Details

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

Examples

```
## Not run:
result <- full_factor("diamonds",c("price","carat","table"))
save_factors(result)
head(dat)

## End(Not run)
```

| | |
|----------------|--|
| 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 glm_reg |
|--------|---|

Details

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

Examples

```
## Not run:
result <- glm_reg("titanic", "survived", "pclass", glm_levels = "Yes")
save_glm_resid(result)

## End(Not run)
```

| | |
|-----------------|--|
| 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 kmeans_clus |
|--------|---|

Details

See 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

```
## Not run:
result <- kmeans_clus("shopping", km_vars = c("v1:v6"))
save_membership(result)

## End(Not run)
```

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

Description

Save regression residuals

Usage

```
save_reg_resid(object)
```

Arguments

| | |
|--------|--|
| object | Return value from regression |
|--------|--|

Details

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

Examples

```
## Not run:  
result <- regression("diamonds", "price", c("carat","clarity"))  
save_reg_resid(result)  
  
## End(Not run)
```

| | |
|-----------|---|
| 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 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://vnijis.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", "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) )
```

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

Details

See 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)
```

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

Details

See 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)
```

| | |
|------------------|--|
| 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 conjoint |
| 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(dataset = "mp3", ca_dep_var = "Rating", ca_indep_var = "Memory:Shape")
summary(result, 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

| | |
|--------|---|
| object | Return value from conjoint_profiles |
| ... | further arguments passed to or from other methods. |

Details

See 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)
```

`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 correlation |
| 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)
```

| | |
|--------------------|---|
| 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 cross_tabs |
| 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 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"))
```

| | |
|---------------------|--|
| 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 full_factor |
| 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 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)
```

| | |
|-----------------|--|
| 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 glm_reg |
| 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 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"))
```

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

Description

Summary method for the hier_clus function

Usage

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

Arguments

| | |
|--------|---|
| object | Return value from hier_clus |
| ... | further arguments passed to or from other methods |

Details

See 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 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)
```

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

Description

Summary method for the mds function

Usage

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

Arguments

| | |
|-----------|--|
| object | Return value from mds |
| 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)
```

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

Description

Summary method for the pmap function

Usage

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

Arguments

| | |
|-------------|---|
| object | Return value from pmap |
| 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)
```

| | |
|--------------------|---|
| 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 pre_factor |
| ... | further arguments passed to or from other methods |

Details

See 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)
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 regression |
| reg_sum_check | Optional output or estimation parameters. "rmse" to show the root mean squared error. "sumsquares" to show the sum of squares table. "vif" to show multi-collinearity diagnostics. "confint" to show coefficient confidence interval estimates. |
| 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")
```

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

Description

Summary method for the sample_size function

Usage

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

Arguments

| | |
|--------|---|
| object | Return value from sample_size |
| ... | further arguments passed to or from other methods |

Details

See 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 | <i>Summary method for the sampling function</i> |
|------------------|---|

Description

Summary method for the sampling function

Usage

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

Arguments

| | |
|--------|---|
| object | Return value from sampling |
| ... | 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)
```

| | |
|---------------------|--|
| 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 single_mean |
| ... | further arguments passed to or from other methods |

Details

See 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)
```

| | |
|---------------------|--|
| 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 single_prop |
| ... | further arguments passed to or from other methods |

Details

See 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)
```

| | |
|------------|--|
| test_check | <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_check(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_check("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 <code>_regression_</code> or <code>_glm_</code> |
| cn | Column names for all independent variables in <code>_dat_</code> |
| 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

Visualize data using ggplot2 <http://docs.ggplot2.org/current/>**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 = "")
```

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") |

DetailsSee <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")
```

| | |
|--------------|--------------------------------------|
| 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**

- city, [5](#)
 - computer, [7](#)
 - diamonds, [10](#)
 - mp3, [19](#)
 - newspaper, [19](#)
 - rndnames, [38](#)
 - shopping, [43](#)
 - titanic, [60](#)
 - titanic_pred, [60](#)
 - toothpaste, [61](#)
- ca_the_table, [3](#)
- changedata, [4](#)
- city, [5](#)
- compare_means, [5](#), [20](#), [47](#)
- compare_props, [6](#), [20](#), [21](#), [47](#), [48](#)
- computer, [7](#)
- conjoint, [4](#), [7](#), [21](#), [48](#)
- conjoint_profiles, [8](#), [11](#), [49](#)
- correlation, [9](#), [22](#), [49](#), [50](#)
- cross_tabs, [10](#), [23](#), [50](#)
- diamonds, [10](#)
- ff_design, [11](#)
- full_factor, [11](#), [23](#), [40](#), [51](#)
- getdata, [12](#)
- glm_reg, [13](#), [24–26](#), [34](#), [40](#), [52](#)
- hier_clus, [14](#), [26](#), [52](#)
- kmeans_clus, [15](#), [27](#), [41](#), [53](#)
- kurtosi, [16](#)
- mac_launcher, [16](#)
- mds, [17](#), [28](#), [54](#)
- mergedata, [18](#)
- mp3, [19](#)
- newspaper, [19](#)
- plot.compare_means, [6](#), [20](#), [47](#)
- plot.compare_props, [7](#), [20](#), [48](#)
- plot.conjoint, [4](#), [8](#), [21](#), [48](#)
- plot.correlation, [9](#), [22](#), [50](#)
- plot.cross_tabs, [10](#), [22](#), [50](#)
- plot.full_factor, [12](#), [23](#), [23](#), [51](#)
- plot.glm_predict, [14](#), [24](#), [26](#), [34](#), [52](#)
- plot.glm_reg, [14](#), [24](#), [25](#), [26](#), [34](#), [52](#)
- plot.hier_clus, [14](#), [26](#), [26](#), [53](#)
- plot.kmeans_clus, [15](#), [27](#), [41](#), [53](#)
- plot.mds, [17](#), [27](#), [54](#)
- plot.pmap, [28](#), [34](#), [55](#)
- plot.pre_factor, [29](#), [36](#), [55](#)
- plot.reg_predict, [31](#)
- plot.regression, [30](#), [31](#), [35](#), [37](#), [56](#)
- plot.single_mean, [32](#), [44](#), [58](#)
- plot.single_prop, [32](#), [45](#), [59](#)
- pmap, [28](#), [29](#), [33](#), [55](#)
- pre_factor, [29](#), [36](#), [55](#)
- predict.glm_reg, [14](#), [24](#), [26](#), [34](#), [52](#)
- predict.regression, [30](#), [31](#), [35](#), [37](#), [56](#)
- radiant, [36](#)
- radiant-package (radiant), [36](#)
- regression, [30](#), [31](#), [35](#), [37](#), [42](#), [56](#)
- rndnames, [38](#)
- sample_size, [38](#), [57](#)
- sampling, [39](#), [57](#), [58](#)
- save_factors, [40](#)
- save_glm_resid, [40](#)
- save_membership, [15](#), [27](#), [41](#), [53](#)
- save_reg_resid, [42](#)
- set_class, [42](#)
- shopping, [43](#)
- sig_stars, [43](#)
- single_mean, [32](#), [44](#), [58](#)
- single_prop, [33](#), [45](#), [59](#)
- skew, [45](#)
- sshh, [46](#)
- sshr, [46](#)
- summary.compare_means, [6](#), [20](#), [47](#)
- summary.compare_props, [7](#), [21](#), [47](#)
- summary.conjoint, [4](#), [8](#), [21](#), [48](#)
- summary.conjoint_profiles, [8](#), [11](#), [49](#)
- summary.correlation, [9](#), [22](#), [49](#)

summary.cross_tabs, [10](#), [23](#), [50](#)
summary.full_factor, [12](#), [51](#)
summary.glm_reg, [14](#), [24](#), [34](#), [51](#)
summary.hier_clus, [14](#), [26](#), [52](#), [53](#)
summary.kmeans_clus, [15](#), [27](#), [41](#), [53](#)
summary.mds, [17](#), [28](#), [54](#)
summary.pmap, [29](#), [34](#), [54](#)
summary.pre_factor, [29](#), [36](#), [55](#)
summary.regression, [30](#), [31](#), [35](#), [37](#), [56](#)
summary.sample_size, [39](#), [57](#)
summary.sampling, [39](#), [57](#)
summary.single_mean, [32](#), [44](#), [58](#)
summary.single_prop, [33](#), [45](#), [59](#)

test_check, [59](#)
titanic, [60](#)
titanic_pred, [60](#)
toothpaste, [61](#)

var_check, [61](#)
visualize, [62](#)

win_launcher, [63](#)