# Package 'BiVariAn'

May 6, 2026

**Type** Package

**Title** Bivariate Automatic Analysis

**Version** 1.0.3

**Date** 2026-01-27

**Maintainer** José Andrés Flores-García <andres.flores@uaslp.mx>

**Description** Simplify bivariate and regression analyses by automating result generation, including summary tables, statistical tests, and customizable graphs. It supports tests for continuous and dichotomous data, as well as stepwise regression for linear, logistic, and Firth penalized logistic models. While not a substitute for tailored analysis, 'BiVariAn' accelerates workflows and is expanding features like multilingual interpretations of results.The methods for selecting significant statistical tests, as well as the predictor selection in prediction functions, can be referenced in the works of Marc Kery (2003) <doi:10.1890/0012-9623(2003)84[92:NORDIG]2.0.CO;2> and Rainer Puhr (2017) <doi:10.1002/sim.7273>.

**License** GPL (>= 3)

**URL** https://github.com/AndresFloresG/BiVariAn

**BugReports** https://github.com/AndresFloresG/BiVariAn/issues

**Imports** car, DescTools, dplyr, epitools, fastDummies, ggplot2, ggprism, glue, lifecycle, logistf, magrittr, rlang, rrtable, scales, stats, systemfonts, table1, tidyr, utils

**Suggests** knitr, riskCommunicator, rmarkdown, spelling, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** José Andrés Flores-García [cre, aut, cph] (ORCID:
<https://orcid.org/0000-0003-0688-3134>),
Antonio Augusto Gordillo-Moscoso [aut] (ORCID:
<https://orcid.org/0000-0002-7351-4614>),
Jhoselin Marian Castro-Rodriguez [aut] (ORCID:
<https://orcid.org/0009-0001-0855-4353>)

**Repository** CRAN

**Date/Publication** 2026-01-27 19:30:02 UTC

# Contents



---

`auto_bar_categ` *Automatic generation of barplot with percentages*

---

## Description

Automatically generates barplot stratified by group variables with or without percentages.

## Usage

```
auto_bar_categ(
  data,
  groupvar = NULL,
  bar_args = NULL,
  theme_func = theme_serene,
  lang_labs = NULL,
  showpercent = TRUE
)
```

## Arguments

| | |
|---|---|
| data | Name of the dataframe |
| groupvar | Name of the grouping variable. Grouping variable will be used in "fill" for aesthetics argument in the creation of each ggplot object. If not provided, the function take each variable as grouping and does not display the "fill" legend. |
| bar_args | List of arguments to be passed to "geom_bar". If NULL, the function uses default arguments such as:<br>• position = "dodge"<br>• colour = "black"<br>• linewidth = 0.9<br>• alpha = 0.5 |
| theme_func | Theme of the generated plots. Must be the name of the function without parenthesis. Use for example: theme_minimal instead of theme_minimal() |
| lang_labs | Language of displayed labels. If null, default is spanish. |
| showpercent | Logical atribute to indicate if the graph should include percentages |

## Value

Returns a list containing all barplots as ggplot object. Can be accessed via $ operator

## Examples

```
data<-data.frame(categ = rep(letters[1:2], 10),
var1 = rep(LETTERS[4:5], 10),
var2 = rep(LETTERS[6:7], 10),
var3 = rep(LETTERS[8:9], 10),
var4 = rep(LETTERS[10:11], 10))

data$categ <- as.factor(data$categ)
data$var1 <- as.factor(data$var1)
data$var2 <- as.factor(data$var2)
data$var3 <- as.factor(data$var3)
data$var4 <- as.factor(data$var4)

barplot_list<-auto_bar_categ(data = data, groupvar = "categ", lang_labs = "EN")

barplot_list$var1
```

```
# Example using `groupvar` argument as `NULL`
auto_bar_categ(data = data)$var2
```

---

auto_bar_cont *Automatic barplot of continous variables*

---

## Description

Generates bar plots of contiuous variables based on numerical variables from a data frame. Internally, the function creates a tibble to summarize the data from each variable.

## Usage

```
auto_bar_cont(
  data,
  groupvar,
  err_bar_show = TRUE,
  err_bar = c("sd", "se"),
  col_args = list(),
  lang_labs = c("EN", "SPA"),
  theme_func = theme_serene
)
```

## Arguments

| | |
|---|---|
| data | Name of the dataframe |
| groupvar | Grouping variable |
| err_bar_show | Logical indicator. Default TRUE show error bars in columns. Default is TRUE |
| err_bar | Statistic to be shown as error bar. Can be "sd" for standard deviation or "se" for standard error. Defauult is "se". |
| col_args | Arguments to be passed to `geom_col` inside the function. Default arguments are:<br>• fill="grey"<br>• color = "black"<br>• alpha = 0.8 |
| lang_labs | Language of the resulting plots. Can be "EN" for english or "SPA" for spanish. Default is "SPA" |
| theme_func | Theme of the generated plots. Must be the name of the function without parenthesis. Use for example: `theme_minimal` instead of `theme_minimal()` |

## Value

Returns a list containing barplots as ggplot2 objects. Objects can be accessed via $ operator.

## Examples

```
data <- data.frame(group = rep(letters[1:2], 30),
var1 = rnorm(30, mean = 15, sd = 5),
var2 = rnorm(30, mean = 20, sd = 2),
var3 = rnorm(30, mean = 10, sd = 1),
var4 = rnorm(30, mean = 5, sd =2))

data$group<-as.factor(data$group)

# Create a list containing all the plots
barcontplots<-auto_bar_cont(data = data, groupvar = 'group', err_bar = "se", lang_labs = 'EN')

# call to show all storaged plots
barcontplots

# call to show one individual plots
barcontplots$var1
```

---

`auto_bp_cont` *auto_bp_cont*

---

## Description

Automatically generates boxplot plots of continuous variables from a database and a grouping variable. The names of the variables are set to the names defined in the database. As a result, graphs generated with the default theme "theme_serene" will be obtained. In this function, the user must define each variable label with "label" function from "table1" package.

## Usage

```
auto_bp_cont(
  data,
  groupvar,
  boxplot_args = list(),
  theme_func = theme_serene,
  lang_labs = c("EN", "SPA")
)
```

## Arguments

| | |
|---|---|
| `data` | Name of the dataframe |
| `groupvar` | Name of the grouping variable |
| `boxplot_args` | List of arguments to be passed to "geom_bar" |
| `theme_func` | Theme to display plots. Default is "theme_serene" |
| `lang_labs` | Language of the resulting plots. Can be "EN" for english or "SPA" for spanish. Default is "SPA" |

## Value

A list containing ggplot2 objects with generated plots. Each element can be accessed by using $ operator.

## Author(s)

JMCR

## Examples

```
data <- data.frame(group = rep(letters[1:2], 30),
var1 = rnorm(30, mean = 15, sd = 5),
var2 = rnorm(30, mean = 20, sd = 2),
var3 = rnorm(30, mean = 10, sd = 1),
var4 = rnorm(30, mean = 5, sd =2))

data$group<-as.factor(data$group)

# Create a list containing all the plots
boxplots<-auto_bp_cont(data = data, groupvar = 'group', lang_labs = 'EN')

# call to show all storaged plots
boxplots

# call to show one individual plots
boxplots$var1
```

---

# auto_corr_cont *Generates automatic scatterplot with correlation plot*

---

## Description

Automatically generates correlation plots of continuous variables from a database and a reference variable. The names of the variables are set to the names defined in the database. As a result, graphs generated with the default theme "theme_serene" will be obtained. In this function, the user must define each variable label with "label" function from "table1" package

## Usage

```
auto_corr_cont(
  data,
  referencevar = NULL,
  point_args = list(),
  smooth_args = list(),
  theme_func = theme_serene,
  lang_labs = c("EN", "SPA")
)
```

### Arguments

| | |
|---|---|
| data | Dataframe from which variables will be extracted |
| referencevar | Reference variable. Must be continuous variable as string (quoted) |
| point_args | List containing extra arguments to be passed to geom_point function. If no specified, only "stat="identity"" will be passed |
| smooth_args | List containing extra arguments to be passed to geom_smooth function. If no specified, only "method="lm"" will be passed |
| theme_func | Theme to display plots. Default is "theme_serene" |
| lang_labs | Language to display title lab. Default is Spanish. |

### Value

Returns a list containing barplots as ggplot2 objects. Objects can be accessed via $ operator.

### Author(s)

JMCR

### Examples

```
data <- data.frame(group = rep(letters[1:2], 30),
var1 = rnorm(30, mean = 15, sd = 5),
var2 = rnorm(30, mean = 20, sd = 2),
var3 = rnorm(30, mean = 10, sd = 1),
var4 = rnorm(30, mean = 5, sd =2))

cont_corrplot <- auto_corr_cont(data = data, referencevar = "var1", lang_labs = "EN")

# Call to show all storaged plots
cont_corrplot

# Call to show one individual plot
cont_corrplot$var2
```

---

## auto_dens_cont *auto_dens_cont*

---

### Description

#' Automatically generates density plots of continuous variables from a database. The names of the variables are set to the names defined in the database. As a result, graphs generated with the default theme "theme_serene" will be obtained. In this function, the user must define each variable label with "label" function from "table1" package.

## Usage

```
auto_dens_cont(
  data,
  s_mean = TRUE,
  s_median = TRUE,
  mean_line_args = list(),
  median_line_args = list(),
  densplot_args = list(),
  theme_func = theme_serene,
  lang_labs = c("EN", "SPA")
)
```

## Arguments

| | |
|---|---|
| `data` | Name of the dataframe |
| `s_mean` | Show mean. Logical operator to indicate if the mean should be plotted. Default is TRUE |
| `s_median` | Show median. Logical operator to indicate if the median should be plotted. Default is TRUE |
| `mean_line_args` | Arguments to be passed to `geom_vline()` of plotted median line when `s_mean = TRUE`. Default arguments are:<br>• color = "red"<br>• linetype="solid"<br>• linewidth = 1 |
| `median_line_args` | Arguments to be passed to `geom_vline()` of plotted median line when `s_median = TRUE`. Default arguments are:<br>• color = "blue"<br>• linetype = "dotdash"<br>• linewidth = 1 |
| `densplot_args` | List of arguments to be passed to "geom_density" |
| `theme_func` | Theme to display plots. Default is "theme_serene" |
| `lang_labs` | Language of the resulting plots. Can be "EN" for english or "SPA" for spanish. Default is "SPA" |

## Value

Returns a list containing the generated density plots

## Author(s)

JMCR

## Examples

```
data <- data.frame(group = rep(letters[1:2], 30),
var1 = rnorm(30, mean = 15, sd = 5),
var2 = rnorm(30, mean = 20, sd = 2),
var3 = rnorm(30, mean = 10, sd = 1),
var4 = rnorm(30, mean = 5, sd =2))

data$group<-as.factor(data$group)

densityplots <- auto_dens_cont(data = data)

densityplots

densityplots$var1
```

---

`auto_pie_categ` *Automatic generation of pieplots*

---

## Description

Generates pie plots based on categorical variables of a data frame.

## Usage

```
auto_pie_categ(
  data,
  pie_bar_args = list(),
  theme_func = theme_serene_void,
  lang_labs = c("EN", "SPA"),
  statistics = TRUE,
  stat_lab = c("percent", "freq"),
  fill_grey = TRUE
)
```

## Arguments

| | |
|---|---|
| `data` | Name of the dataframe |
| `pie_bar_args` | List of arguments to be passed to "geom_bar" |
| `theme_func` | Theme of the generated plots. Default is "theme_serene_void" |
| `lang_labs` | Language of displayed labels. If null, default is spanish. |
| `statistics` | Logical attribute to indicate if summary statistic parameters are shown. |
| `stat_lab` | Statistics to be shown. Can choose if you want to show percentages or frequencies. |
| `fill_grey` | Logical indicator to choose if the generated pie plots must be grey. Default is TRUE. |

## Value

Returns a list containing barplots as ggplot2 objects. Objects can be accessed via $ operator.

## Examples

```
data <- data.frame(categ = rep(c("Categ1", "Categ2"), 25),
var1 = rbinom(50, 2, prob = 0.3),
var2 = rbinom(50, 2, prob = 0.8),
var3 = rbinom(50, 2, prob = 0.7))
data$categ <- as.factor(data$categ)
data$var1 <- as.factor(data$var1)
data$var2 <- as.factor(data$var2)
data$var3 <- as.factor(data$var3)

pieplot_list <- auto_pie_categ(data = data)

# Call for all listed plots
pieplot_list

# Call for one specific plot
pieplot_list$var1
```

---

# auto_shapiro_raw *Automatic Shapiro-Wilk test table*

---

## Description

Generates a HTML table of raw data from a numerical variables of a dataframe.

## Usage

```
auto_shapiro_raw(data, flextableformat = TRUE)
```

## Arguments

| | |
|---|---|
| data | Data frame from which variables will be extracted. |
| flextableformat | Logical operator to indicate the output desired. Default is TRUE. When FALSE, function will return a dataframe format. |

## Value

Flextable or dataframe with shapiro wilks results.

## Author(s)

JAFG

## Examples

```
auto_shapiro_raw(iris)
```

---

`auto_viol_cont` *auto_viol_cont*

---

## Description

Automatically generates violinplots of continuous variables from a database and a grouping variable. The names of the variables are set to the names defined in the database. As a result, graphs generated with the default theme "theme_serene" will be obtained. In this function it is not possible to use labels for the variables, use "auto_viol_cont_wlabels" instead.

## Usage

```
auto_viol_cont(
  data,
  groupvar,
  violinplot_args = list(),
  theme_func = theme_serene,
  lang_labs = c("EN", "SPA")
)
```

## Arguments

| | |
|---|---|
| `data` | Name of the dataframe |
| `groupvar` | Name of the grouping variable |
| `violinplot_args` | List of arguments to be passed to "geom_violin" |
| `theme_func` | Theme to display plots. Default is "theme_serene" |
| `lang_labs` | Language of the resulting plots. Can be "EN" for english or "SPA" for spanish. Default is "SPA". |

## Value

Returns a list containing barplots as ggplot2 objects. Objects can be accessed via $ operator.

## Author(s)

JMCR

## Examples

```
data <- data.frame(group = rep(letters[1:2], 30),
var1 = rnorm(30, mean = 15, sd = 5),
var2 = rnorm(30, mean = 20, sd = 2),
var3 = rnorm(30, mean = 10, sd = 1),
var4 = rnorm(30, mean = 5, sd =2))

data$group<-as.factor(data$group)

# Create a list containing all the plots
violinplots<-auto_viol_cont(data = data, groupvar = 'group', lang_labs = 'EN')

# call to show all storaged plots
violinplots

# call to show one individual plots
violinplots$var1
```

---

# continuous_2g *Bivariate analysis for 2 groups*

---

## Description

Automatic test for continuous variables for 2 groups. Variable names can be assigned using `table1::label()` function.

## Usage

```
continuous_2g(
  data,
  groupvar,
  ttest_args = list(),
  wilcox_args = list(),
  flextableformat = TRUE
)
```

## Arguments

| | |
|---|---|
| `data` | Data frame from which variables will be extracted. |
| `groupvar` | Grouping variable as character. Must have exactly 2 levels. |
| `ttest_args` | Arguments to be passed to `t.test()` function. |
| `wilcox_args` | Arguments to be passed to `wilcox.test()` function. |
| `flextableformat` | Logical operator to indicate the output desired. Default is TRUE. When FALSE, function will return a dataframe format. |

## Value

Returns a dataframe or flextable of 2 groups 2 sided Mann Whitney's U or T test, along with Shapiro-Wilk's p values and Levene's p value.

## Examples

```
df <- mtcars
df$am <- as.factor(df$am)
continuous_2g(data = df,
groupvar = "am",
flextableformat = FALSE)

# Set names to variables
if(requireNamespace("table1")){
table1::label(df$mpg) <- "Miles per gallon"
table1::label(df$cyl) <- "Number of cylinders"
table1::label(df$disp) <- "Displacement"
table1::label(df$hp) <- "Gross horsepower"
table1::label(df$drat) <- "Rear axle ratio"

continuous_2g(data = df, groupvar = "am", flextableformat = FALSE)
}
```

---

# continuous_2g_pair *Bivariate analysis for 2 groups for paired data*

---

## Description

Automatic paired test for continuous variables for 2 groups. Variable names can be assigned using `table1::label()` function.

## Usage

```
continuous_2g_pair(
  data,
  groupvar,
  ttest_args = list(),
  wilcox_args = list(),
  flextableformat = TRUE
)
```

## Arguments

| | |
|---|---|
| `data` | Data frame from which variables will be extracted. |
| `groupvar` | Grouping variable. Must have exactly 2 levels. |
| `ttest_args` | Arguments to be passed to `t.test()` function. |
| `wilcox_args` | Arguments to be passed to `wilcox.test()` function. |

`flextableformat`
: Logical operator to indicate the output desired. Default is TRUE. When FALSE, function will return a dataframe format.

## Value

A dataframe or flextable with containing p values for paired tests along with statistics for normality and homocedasticity.

## Examples

```
data <- data.frame(group = rep(letters[1:2], 30),
                   var1 = rnorm(60, mean = 15, sd = 5),
                  var2 = rnorm(60, mean = 20, sd = 2),
                  var3 = rnorm(60, mean = 10, sd = 1),
                  var4 = rnorm(60, mean = 5, sd =2))
data$group<-as.factor(data$group)

continuous_2g_pair(data = data, groupvar = "group")

# Set names to variables
if(requireNamespace("table1")){
table1::label(data$var1) <- "Variable 1"
table1::label(data$var2) <- "Variable 2"
table1::label(data$var3) <- "Variable 3"
table1::label(data$var4) <- "Variable 4"

continuous_2g_pair(data = data, groupvar = "group", flextableformat = FALSE)
}
```

---

# continuous_corr_test *Bivariate analysis for correlation tests*

---

## Description

Automatic correlation analyses for continuous variables with one variable as reference. Variable names can be assigned using `table1::label()` function.

## Usage

```
continuous_corr_test(
  data,
  referencevar,
  alternative = NULL,
  flextableformat = TRUE,
  corr_test = c("all", "pearson", "spearman", "kendall")
)
```

## Arguments

| | |
|---|---|
| data | Data frame from which variables will be extracted. |
| referencevar | Reference variable. Must be a continuous variable. |
| alternative | Alternative for cor.test. Must be either "two.sided", "geater" or "less" |
| flextableformat | Logical operator to indicate the output desired. Default is TRUE. When FALSE, function will return a dataframe format. Because the function calculates different statistics for each correlation (specially in kendall correlation test), it may take some time to run. You can select individual variables using the pipe operator and the select function to run correlations only on the selected variables. |
| corr_test | Correlation test to be performed |

## Value

A dataframe or flextable containing pvalues for correlation tests along with the normality and homocedasticity tests p values

## Examples

```
# example code

data <- data.frame(group = rep(letters[1:2], 15),
var1 = rnorm(30, mean = 15, sd = 5),
var2 = rnorm(30, mean = 20, sd = 2),
var3 = rnorm(30, mean = 10, sd = 1),
var4 = rnorm(30, mean = 5, sd =2))

data$group<-as.factor(data$group)

continuous_corr_test(data = data, referencevar = "var1", flextableformat = FALSE)

# Set names to variables
if(requireNamespace("table1")){
table1::label(data$var2) <- "Variable 2"
table1::label(data$var3) <- "Variable 3"
table1::label(data$var4) <- "Variable 4"

continuous_corr_test(data = data, referencevar = "var1", flextableformat = FALSE)
}

# Example performing correlation test for only one variable
if(requireNamespace("dplyr")){
library(dplyr)
continuous_corr_test(data = data %>% select("var1","var2"),
 referencevar = "var1", flextableformat = FALSE, corr_test = "pearson")
}

# Example performing only pearson correlation test
continuous_corr_test(data = data, referencevar = "var1",
 flextableformat = FALSE, corr_test = "pearson")
```

`continuous_multg` *Bivariate analysis for more than 2 groups*

## Description

Generates a HTML table of bivariate analysis for 2 groups.

## Usage

```
continuous_multg(data, groupvar, flextableformat = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | Data frame from which variables will be extracted. |
| `groupvar` | Grouping variable. Must have exactly 2 levels. |
| `flextableformat` | Logical operator to indicate the output desired. Default is TRUE. When FALSE, function will return a dataframe format. |

## Value

A dataframe or flextable containing pvalues for each test along with the normality and homocedasticity tests p values. An extra column will be shown indicating the recommended significant test

## Examples

```
data <- iris

data$Species<-as.factor(data$Species)

continuous_multg(data = data, groupvar = "Species", flextableformat = FALSE)
```

dichotomous_2k_2sid *Bivariate Chi squared and Fisher Test analysis for 2 categories.*

## Description

Generates a HTML table of bivariate Chi squared and Fisher Test analysis for 2 categories. Display a table arranged dataframe with Chi squared statistic, minimum expected frecuencies, Chi squared p value, Fisher Test p value, and Odds ratio with 95 confidence levels. Note that you must recode factors and level the database factors in order to compute exact p values. Variable names can be assigned using `table1::label()` function.

## Usage

```
dichotomous_2k_2sid(data, referencevar, flextableformat = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | Data frame from which variables will be extractred |
| `referencevar` | Reference variable. Must have exactly 2 levels |
| `flextableformat` | Logical operator to indicate the output desired. Default is TRUE. When FALSE, function will return a dataframe format. |

## Value

Returns a dataframe or flextable containing statistical values for Chi squared tests or Fisher's test.

## Author(s)

JAFG

## Examples

```
 # Not run

# Create a sample dataframe
df <- data.frame(
  has = c("Yes", "No", "Yes", "Yes", "No", "No", "Yes"),
  smoke = c("Yes", "No", "No", "Yes", "No", "Yes", "No"),
  gender = c("Male", "Female", "Male", "Female", "Female", "Male", "Male"))

df$has <- as.factor(df$has)
df$smoke <- as.factor(df$smoke)
df$gender <- as.factor(df$gender)

# Set a value as reference level
df$has <- relevel(df$has, ref= "Yes")
df$smoke <- relevel(df$smoke, ref= "Yes")
```

```
 df$gender <- relevel(df$gender, ref= "Female")

 # Apply function
dichotomous_2k_2sid(df, referencevar="has")
dichotomous_2k_2sid(df, referencevar="has", flextableformat = FALSE)

# Set names to variables
if(requireNamespace("table1")){
table1::label(df$has) <- "Hypertension"
table1::label(df$smoke) <- "Smoking Habits"
table1::label(df$gender) <- "Gender"

dichotomous_2k_2sid(df, referencevar="has", flextableformat = FALSE)
}
```

---

`encode_factors` *Encode character variables as factor automatically*

---

## Description

Encode character variables as factor automatically

## Usage

```
encode_factors(
  data,
  encode = c("character", "integer"),
  list_factors = NULL,
  uselist = FALSE
)
```

## Arguments

| | |
|---|---|
| `data` | Dataframe to be encoded |
| `encode` | Column class to be encoded. Must be "character" or "integer" |
| `list_factors` | List of factors to be encoded |
| `uselist` | Logical operator to determine if use list of factors or not. If TRUE, list_factors argument must be provided. |

## Value

Converts listed columns to factors.

## Examples

```
df <- data.frame(has = c("Yes", "No", "Yes", "Yes", "No", "No", "Yes"),
smoke = c("Yes", "No", "No", "Yes", "No", "Yes", "No"),
gender = c("Male", "Female", "Male", "Female", "Female", "Male", "Male"))

str(df)

df <- encode_factors(df, encode = "character")

str(df)
```

---

# logistf_summary *Summary method for logistf with no printable output*

---

## Description

Summary method for logistf models, currently this method is only used in step_bw_firth function.

## Usage

```
logistf_summary(object, verbose = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | logistf class object |
| verbose | logical. If TRUE, the output will be printed |
| ... | Additional arguments |

## Value

An object class 'data.frame' showing coefficients and p_values.

## References

Heinze G, Ploner M, Jiricka L, Steiner G. logistf: Firth's Bias-Reduced Logistic Regression. 2023. available on: https://CRAN.R-project.org/package=logistf

## Examples

```
# Only use if you want a non-printable version of 'summary' for a logistfnp object.
if (requireNamespace("logistf")) {
  library(logistf)
  data <- mtcars
  data$am <- as.factor(data$am)

  regression_model <- logistf::logistf(am ~ mpg + cyl + disp, data = data)
```

```
  logistf_summary(regression_model)
}
```

---

ss_multreg *Sample Size Calculation for multiple regression analysis*

---

## Description

Calculates the recommended sample size for a multiple regression analysis.

## Usage

```
ss_multreg(df, prop = NULL, logistic = FALSE, verbose = TRUE)
```

## Arguments

| | |
|---|---|
| df | Degrees of freedom planned to be introduced |
| prop | Minimum prevalence of the expected event (Required if planned regression is a logistic regression) |
| logistic | Logical operator to indicate wether the planned regression analysis is a logistic regression or not. |
| verbose | Logical operator to indicate wether the results should be printed in console. Default is TRUE |

## Value

An object class ss_multreg_obj indicating the sample size calculation for a regression analysis.

## References

Peduzzi P, Concato J, Kemper E, Holford TR, Feinstein AR. A simulation study of the number of events per variable in logistic regression analysis. Journal of Clinical Epidemiology. diciembre de 1996;49(12):1373–9.

Pierdant-Pérez M, Patiño-López MI, Flores-García JA, Jacques-García FA. Implementación de un curso virtual de lectura crítica en estudiantes de medicina durante la pandemia COVID-19. Inv Ed Med. el 1 de octubre de 2023;12(48):64–71.

## Examples

```
# Lineal multiple regression with 4 degrees of freedom
ss_multreg(4, logistic = FALSE)

# Logistic multiple regression with 4 degrees of freedom
# and 60% of probability of the event

ss_multreg(4, prop = .6, logistic = TRUE)
```

`step_bw_firth` *Stepwise backward for logistic Firth regression with automated dummy variables conversion (with forced terms)*

## Description

Extension code to perform stepwise backward on a logistf model with categorical variables. Automatically transforms predictors that are factors into dummy variables.

## Usage

```
step_bw_firth(
  reg_model,
  s_lower = "~1",
  s_upper = "all",
  trace = TRUE,
  steps = NULL,
  p_threshold = 0.05,
  data = NULL,
  forced = NULL
)
```

## Arguments

| | |
|---|---|
| `reg_model` | A logistf model object. |
| `s_lower` | Lower step. Not used; included for compatibility. Default = "~1". |
| `s_upper` | Upper step. Not used; included for compatibility. Default = "all". |
| `trace` | Logical. Print each step. Default TRUE. |
| `steps` | Maximum number of elimination steps. If NULL, set to number of predictors. |
| `p_threshold` | P-value threshold for elimination. Default = 0.05. |
| `data` | Data frame. If NULL, extracted from reg_model. |
| `forced` | Character vector of term names to always keep. Default NULL. |

## Value

An object of class "step_bw" with components:

- final_model: the fitted logistf model
- steps: a data.frame of each elimination step

## References

Heinze G, Ploner M, Jiricka L, Steiner G. logistf: Firth's Bias-Reduced Logistic Regression. 2023.

Efroymson MA. Multiple regression analysis. In: Ralston A, Wilf HS, editors. Mathematical methods for digital computers. New York: Wiley; 1960.

Ullmann T, Heinze G, Hafermann L, Schilhart-Wallisch C, Dunkler D, et al. (2024) Evaluating variable selection methods for multivariable regression models: A simulation study protocol. PLOS ONE 19(8): e0308543

## Examples

```
if (requireNamespace("logistf", quietly = TRUE)) {
  library(logistf)
  data <- mtcars
  data$am <- as.factor(data$am)
  regression_model <- logistf(am ~ mpg + cyl + disp, data = data)
  # Perform backward stepwise, forcing 'cyl' to remain
  stepwise <- step_bw_firth(
    regression_model,
    forced = c("cyl"),
    p_threshold = 0.05,
    trace = FALSE
  )
  final_model <- stepwise$final_model
  # Show steps and summary
  stepwise$steps
  summary(final_model)
}
```

---

# step_bw_p *Automatized stepwise backward for regression models (with forced terms)*

---

## Description

Automatized stepwise backward for regression models (with forced terms)

## Usage

```
step_bw_p(
  reg_model,
  s_lower = "~1",
  s_upper = "all",
  trace = TRUE,
  steps = NULL,
  p_threshold = 0.05,
  data = NULL,
```

```
  forced = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| reg_model | Regression model. Must be a glm or lm object |
| s_lower | Lower step. Names of the variables to be included at the lower step. Default is "~1" (intercept only) |
| s_upper | Upper step. Names of the variables to be included at the upper step. Default is "all" (all predictors) |
| trace | Logical. Whether to print each step to the console. Default is TRUE. |
| steps | Maximum number of elimination steps. If NULL, set to number of predictors. |
| p_threshold | P-value threshold for elimination. Default is 0.05. |
| data | Data frame for stepwise. If NULL, extracted from reg_model. |
| forced | Character vector of predictor names to always keep. Default NULL. |
| ... | Additional arguments passed to car::Anova(). |

## Value

An object of class "step_bw" containing the final model and a data.frame of steps.

## References

Efroymson MA. Multiple regression analysis. In: Ralston A, Wilf HS, editors. Mathematical methods for digital computers. New York: Wiley; 1960.

## Examples

```
data(mtcars)
# Fit a linear model
regression_model <- lm(cyl ~ ., data = mtcars)
# Perform backward stepwise, forcing "wt" and "hp" to remain
stepwise <- step_bw_p(
  regression_model,
  forced = c("wt", "hp"),
  trace = FALSE
)
# Extract final model and view summary
final_model <- stepwise$final_model
summary(final_model)
```

theme_serene *Basic theme for Bivaran packages plots*

## Description

Basic theme for Bivaran packages plots

## Usage

```
theme_serene(
  base_size = 14,
  base_family = "sans",
  base_fontface = "plain",
  base_line_size = base_size/14,
  base_rect_size = base_size/14,
  axis_text_angle = 0,
  border = FALSE
)
```

## Arguments

| | |
|---|---|
| base_size | base font size, given in pts. |
| base_family | base font family |
| base_fontface | base font face |
| base_line_size | base line size |
| base_rect_size | base rect size |
| axis_text_angle | Axis text angle |
| border | Logical operator to indicate if the border should be printed |

## Value

Returns a list of classes "gg" and "theme"

## Author(s)

Jhoselin Marian Castro-Rodriguez

## Examples

```
library(ggplot2)
data <- mtcars
p1 <- ggplot(data, aes(disp, hp))+
geom_point()+
geom_smooth()
```

```
p1 + theme_serene()
```

---

`theme_serene_void` *Void theme for Bivaran packages plots*

---

## Description

Basic theme for Bivaran packages plots

## Usage

```
theme_serene_void(
  base_size = 11,
  base_family = "sans",
  base_fontface = "plain",
  base_line_size = base_size/22,
  base_rect_size = base_size/2,
  axis_text_angle = 0,
  border = FALSE
)
```

## Arguments

| | |
|---|---|
| `base_size` | base font size, given in pts. |
| `base_family` | base font family |
| `base_fontface` | base font face |
| `base_line_size` | base line size |
| `base_rect_size` | base rect size |
| `axis_text_angle` | Axis text angle |
| `border` | Logical operator to indicate if the border should be printed |

## Value

Returns a list of classes "gg" and "theme"

## Author(s)

Jhoselin Marian Castro-Rodriguez

## Examples

```
library(ggplot2)

data <- mtcars
p1 <- ggplot(data, aes(disp, hp))+
geom_point()+
geom_smooth()

p1 + theme_serene_void()
```

# Index