R documentation

of all in ‘man’

October 29, 2018

R topics documented:

nt_barplot 2
nt_boxplot 3
nt_compare_mg 4
nt_compare_tg 5
nt_describe 6
nt_dotplot 7
nt_km 8
nt_multiple_cox 9
nt_multiple_logistic 10
nt_norm_test 11
nt_simple_cox 12
nt_simple_logistic 13
put_together 14
ql_var 15
qt_var 16
StatStepribbon 16
stat_stepribbon 17
std_barplot 18
std_barplot_group 18
std_boxplot 19
std_boxplot_group 19
std_dotplot 20
std_dotplot_group 20
std_km 21
std_km_group 21

Index **23**

`nt_barplot` *Barplot*

Description

Plot barplot for several variables.

Usage

```
nt_barplot(data, group = NULL, ylab = "Percent (%)", save = FALSE,
  fig.height = 5, fig.width = 5, std_fun = std_barplot,
  std_fun_group = std_barplot_group)
```

Arguments

`data`	a data frame with the variables.
`group`	an optional data frame with the group variable.
`ylab`	a character value specifying the y axis label.
`save`	a logical value indicating whether the output should be saved as a jpeg file.
`fig.height`	a numeric value indicating the height (in) of the file.
`fig.width`	a numeric value indicating the width (in) of the file.
`std_fun`	a function to plot a barplot when `group = NULL`. It must follow the same structure of `std_barplot`.
`std_fun_group`	a function to plot a dotplot when `group` is provided. It must follow the same structure of `std_barplot_group`.

Details

The functions `std_barplot` and `std_barplot_group` are stardard functions that can be modified by the user in order to customize the barplots a prior. The plots also can be modified a posterior as a regular ggplot object. See `geom_bar`, `std_barplot` and `std_barplot_group`.

Value

a list of ggplot objects with each item named by the column names from `var`.

Examples

```
library(dplyr)
library(magrittr)
data(iris)

vars <- iris %>%
transmute(Species = Species,
        Sepal.Length.C = ifelse(Sepal.Length > 5, "> 5", "<= 5"),
        Sepal.Width.C = ifelse(Sepal.Width > 3.5, "> 3.5", "<= 3.5"))
vars %>% nt_barplot(group = Species)
```

nt_boxplot *Boxplot*

Description

Plot boxplot for several variables.

Usage

```
nt_boxplot(data, group = NULL, save = FALSE, fig.height = 5,
  fig.width = 5, std_fun = std_boxplot,
  std_fun_group = std_boxplot_group)
```

Arguments

`data`	a data frame with the variables.
`group`	an optional data frame with the group variable.
`save`	a logical value indicating whether the output should be saved as a jpeg file.
`fig.height`	a numeric value indicating the height (in) of the file.
`fig.width`	a numeric value indicating the width (in) of the file.
`std_fun`	a function to plot a boxplot when `group = NULL`. It must follow the same structure of `std_boxplot`.
`std_fun_group`	a function to plot a boxplot when group is provided. It must follow the same structure of `std_boxplot_group`.

Details

The functions `std_boxplot` and `std_boxplot_group` can be modified by the user in order to customize the boxplots a prior. The plots also can be modified a posterior as a regular ggplot object. See `geom_boxplot`, `std_boxplot` and `std_boxplot_group`.

Value

a list of ggplot objects with each item named by the column names from `var`.

Examples

```
library(magrittr)
data(iris)

iris %>% nt_boxplot(group = Species)
```

`nt_compare_mg` *Compare more than two groups*

Description

Performing comparisons among three or more groups.

Usage

```
nt_compare_mg(data, group, test = "auto", norm.test = "sf",
  digits.p = 3, save = FALSE, file = "nt_compare_mg", mc = FALSE)
```

Arguments

`data`	a data frame with the variables.
`group`	a data frame with the group variable.
`test`	a character value indicating the tests to be performed. The options are "auto", "par" and "npar". See more in details.
`norm.test`	a character value specifying the normality test to be performed. The options are Anderson-Darling (`ad`), Shapiro-Francia (`"sf"`), Kolmogorov-Smirnov (`ks`), Cramer-vonMises (`cvm`) and Pearson (`p`). The default is Shapiro-Francia (`"sf"`). It is only used if `test = "automatic"`.
`digits.p`	the number of digits to present the p-values.
`save`	a logical value indicating whether the output should be saved as a csv file.
`file`	a character value indicating the name of output file in csv format to be saved.
`mc`	a logical value indicating if pairwise comparisons should will be performed.

Details

If `test = "automatic"`, the normality assumption will be verified by a normality test (Anderson-Daling (ad.test), Shapiro-Francia (sf.test), 'Kolmogorov-Smirnov (lillie.test), Cramer-vonMises (cvm.test), and Pearson (pearson.test)) and Levene test (leveneTest) will evaluate the assumption of homocedasticity at a significance level of 0.05. If the data satisfies both assumptions, then ANOVA is chosen; if only normality is satisfied, then Welch ANOVA; if only homoscedasticity or neither assumptions, then Kruskal-Wallis.

Examples

```
library(magrittr)
data(iris)

iris %>% nt_compare_mg(group = Species)
```

`nt_compare_tg` *Compare two groups*

Description

Performing comparisons between two groups.

Usage

```
nt_compare_tg(data, group, alternative = "two.sided", test = "auto",
  conf.level = 0.95, paired = FALSE, norm.test = "sf",
  format = TRUE, digits.ci = 3, digits.p = 5, save = FALSE,
  file = "nt_compare_tg")
```

Arguments

`data`	a data frame with the variables.
`group`	a data frame with the group variable.
`alternative`	a character value indicating the alternative hypothesis, must be one of "two.sided", "greater" or "less".
`test`	a character value indicating the tests to be performed. The options are "auto", "par", "npar", "t", "wt", "mw", "bm". See more in details.
`conf.level`	a character value specifying the confidence level of the confidence interval for the difference between the two groups.
`paired`	a logical value indicating whether a paired test should be used.
`norm.test`	a character value specifying the normality test to be performed. The options are Anderson-Darling (`ad`), Shapiro-Francia (`"sf"`), Kolmogorov-Smirnov (`ks`), Cramer-vonMises (`cvm`) and Pearson (`p`). The default is Shapiro-Francia (`"sf"`). It is only used if `test = "automatic"`.
`format`	a logical value indicating whether the output should be formatted.
`digits.ci`	the number of digits to present the confidence intervals.
`digits.p`	the number of digits to present the p-values.
`save`	a logical value indicating whether the output should be saved as a csv file.
`file`	a character value indicating the name of output file in csv format to be saved.

Details

If `test = "auto"`, the normality assumption will be verified by a normality test (Anderson-Daling (ad.test), Shapiro-Francia (sf.test), 'Kolmogorov-Smirnov (lillie.test), Cramer-vonMises (cvm.test), or Pearson (pearson.test)) and Levene test (leveneTest) will evaluate the assumption of homocedasticity at a significance level of 0.05. If the data satisfies both assumptions, then t-test is chosen; if only normality is satisfied, then Welch t-test is performed; if only homoscedasticity, then Mann-Whitney is used; if neither assumptions, then Brunner-Munzel t test is applied; If `test = "par"`, then only parametric tests will be performed; If `test = "npar"`, then only non-parametric tests will be performed; If either "t", "wt", "mw" and "bm", then only t-test, Welch-t test, Mann-Whitney or Brunner-Munzel test will be performed, respectively.

Examples

```
library(dplyr)
library(forcats)
data(iris)

iris_nt <- iris %>% filter(Species != "setosa") %>% mutate(Species = fct_drop(Species))
iris_nt %>% nt_compare_tg(group = Species)
```

---

nt_describe *Descriptive measures*

Description

Calculating a descriptive table for quantitative and qualitative variables in the usual format to present in scientific article.

Usage

```
nt_describe(data, group = NULL, measures = c("mean.sd", "median.iqr",
  "median.range", "missing"), digits = 2, save = FALSE,
  file = "descriptive_analysis")
```

Arguments

data	a data frame with the variables.
group	an optional with the group variable.
measures	a character value indicating which measures should be presented: mean.sd, mean.iqr, median.range, and missing.
digits	a numeric value specifying the number of digits to present the results.
save	a logical value indicating whether the output should be saved as a csv file.
file	a character indicating the name of output file in csv format to be saved.

Value

A data frame with mean, standard deviation, median, quantile 25%, quantile 75%, minimum, maximum, sample size and missing data for quantitative variables, and frequency, percentage, sample size and missing data for qualitative variables.

Examples

```
library(dplyr)
library(magrittr)
data(iris)

iris_nt <- iris %>%
 mutate(Species = ql_var(Species,
                        from = c("setosa", "versicolor", "virginica"),
                        to = c("Setosa", "Versicolor", "Virginica"),
                        order = c("Virginica", "Setosa", "Versicolor")))
iris_nt %>% nt_describe(group = Species)
```

`nt_dotplot` *Dotplot*

Description

Plot dotplot for several variables.

Usage

```
nt_dotplot(data, group = NULL, binwidth = 1, save = FALSE,
  fig.height = 5, fig.width = 5, std_fun = std_dotplot,
  std_fun_group = std_dotplot_group)
```

Arguments

`data`	a data frame with the variables.
`group`	an optional data frame with the group variable.
`binwidth`	a numerical vector specifying the bin width for each variable.
`save`	a logical value indicating whether the output should be saved as a jpeg file.
`fig.height`	a numeric value indicating the height (in) of the file.
`fig.width`	a numeric value indicating the width (in) of the file.
`std_fun`	a function to plot a dotplot when `group = NULL`. It must follow the same structure of the function `std_dotplot`.
`std_fun_group`	a function to plot a dotplot when `group` is provided. It must follow the same structure of the function `std_dotplot_group`.

Details

The functions std_dotplot and std_dotplot_group can be modified by the user in order to customize the dotplots a prior. The plots also can be modified a posterior as a regular ggplot object. See `geom_dotplot`, `std_dotplot` and `std_dotplot_group`.

Value

a list of ggplot objects with each item named by the column names from `var`.

Examples

```
library(magrittr)
data(iris)

iris %>% nt_dotplot(group = Species, binwidth = 0.1)
```

`nt_km` *Kaplan-Meier plot*

Description

Plot Kaplan-Meier curves for several variables.

Usage

```
nt_km(data, time, status, xlab = "Time", ylab = "Survival",
  save = FALSE, fig.height = 5, fig.width = 5, std_fun = std_km,
  std_fun_group = std_km_group, time.points = NULL, format = TRUE,
  digits = 2, file = "survival")
```

Arguments

`data`	a data frame with the variables.
`time`	a data frame with the time to event variable.
`status`	a data frame with the indicator associated to events.
`xlab`	a character value specifying the x axis label.
`ylab`	a character value specifying the y axis label.
`save`	a logical value indicating whether the output should be saved as a jpeg file.
`fig.height`	a numeric value indicating the height (in) of the file.
`fig.width`	a numeric value indicating the width (in) of the file.
`std_fun`	a function to plot a barplot when `group = NULL`. It must follow the same structure of `std_barplot`.
`std_fun_group`	a function to plot a dotplot when `group` is provided. It must follow the same structure of `std_barplot_group`.
`time.points`	a numeric vector of time points to evaluate the survival curves.
`format`	a logical value indicating whether the output should be formatted.
`digits`	a numerical value defining of digits to present the results.
`file`	a character indicating the name of output file in csv format to be saved.

Details

The functions `std_km` and `std_km_group` are standard functions that can be modified by the user in order to customize the barplots a prior. The plots also can be modified a posterior as a regular ggplot object. See `std_km` and `std_km_group`.

Value

a list of ggplot objects with each item named by the column names from `var`.

Examples

```
library(survival)
data(lung)

lung_nt <- lung %>% mutate(sex = ql_var(sex, from = 1:2,
                                        to = c("Female", "Male"),
                                        label = "Sex"),
                       ph.ecog = ql_var(ph.ecog, label = "ECOG")) %>%
                       select(sex, ph.ecog, time, status)
lung_nt %>% nt_km(time = time, status = status)
```

`nt_multiple_cox` *Proportional Hazards Cox regression table*

Description

Tabulating results from fitted Proportional Hazards Cox models.

Usage

```
nt_multiple_cox(fit.list, fit.labels = NULL, ci.type = "lr",
  tab.type = "hr", format = TRUE, digits = 2, digits.p = 3,
  save = FALSE, file = "nt_multiple_cox")
```

Arguments

`fit.list`	a list of fitted models.
`fit.labels`	a character vector labelling the models in `fit.list`
`tab.type`	a character value indicating the procedure to calculate confidence intervals: likelihood ratio (`lr`) or wald (`wald`).
`format`	a logical value indicating whether the output should be formatted.
`digits`	a numerical value defining of digits to present the results.
`digits.p`	a numerical value defining number of digits to present the p-values.
`save`	a logical value indicating whether the output should be saved as a csv file.
`file`	a character indicating the name of output file in csv format to be saved.
`tab.type`	a character value indicating either hazard ratio (`hr`) or coefficients (`coef`) will be shown in the output.

Examples

```
library(survival)
library(magrittr)
library(dplyr)

data(ovarian)
ovarian_nt <- ovarian %>% mutate(resid.ds = ql_var(resid.ds,
                                                   from = 1:2,
                                                   to = c("no", "yes"),
                                                   label = "Residual Disease"),
                              ecog.ps = ql_var(ecog.ps,
                                               from = 1:2,
                                               to = c("I", "II"),
                                               label = "ECOG-PS"),
                              rx = ql_var(rx,
                                          from = 1:2,
                                          to = c("t1", "t2"),
                                          label = "Treatment"),
                              age = qt_var(age,
                                           label = "Age"))

fit.list <- list()

fit.list[[1]] <- coxph(Surv(futime, fustat) ~ ecog.ps, data = ovarian_nt)
fit.list[[2]] <- coxph(Surv(futime, fustat) ~ resid.ds + rx, data = ovarian_nt)
fit.list[[3]] <- coxph(Surv(futime, fustat) ~ age + ecog.ps*rx, data = ovarian_nt)

nt_multiple_cox(fit.list)
```

nt_multiple_logistic *Logistic regression table*

Description

Tabulating results from Logistic models.

Usage

```
nt_multiple_logistic(fit.list, fit.labels = NULL, ci.type = "lr",
  tab.type = "or", format = TRUE, digits = 2, digits.p = 3,
  save = FALSE, file = "nt_multiple_logistic")
```

Arguments

`fit.list`	a list of fitted models.
`fit.labels`	a character vector labelling the models in `fit.list`
`format`	a logical value indicating whether the output should be formatted.
`digits`	a numerical value defining of digits to present the results.
`digits.p`	a numerical value defining number of digits to present the p-values.
`save`	a logical value indicating whether the output should be saved as a csv file.
`file`	a character indicating the name of output file in csv format to be saved.
`type`	a character value indicating either odds ratio (`or`) or coefficients (`coef`) will be shown in the output.

Examples

```
library(titanic)
library(magrittr)
library(dplyr)

data(titanic_train)
dt <- titanic_train %>% mutate(Sex = ql_var(Sex,
                                            from = c("male", "female"),
                                            to = c("Male", "Female")),
                               Pclass = ql_var(Pclass,
                                               from = 1:3,
                                               to = c("I", "II", "III"),
                                               label = "Passenger Class"))

fit.list <- list()

fit.list[[1]] <- glm(Survived ~ Sex, data = dt)
fit.list[[2]] <- glm(Survived ~ Age + Sex, data = dt)
fit.list[[3]] <- glm(Survived ~ Age + Sex + Pclass, data = dt)

nt_multiple_logistic(fit.list)
```

`nt_norm_test` *Normality tests*

Description

Perform Anderson-Darling, Shapiro-Francia, Kolmogorov-Smirnov, Cramer-vonMises, and Pearson normality tests for several variables. In addition, it also plots a Quantile-Quantile plot and a histogram.

Usage

```
nt_norm_test(data, group = NULL, test = "sf", digits = 3,
  pvalue.plot = TRUE)
```

Arguments

`data`	a data frame with the variables.
`group`	an optional character indicating the group variable.
`test`	a character value specifying the normality test to be performed. The options are Anderson-Darling (`ad`), Shapiro-Francia (`"sf"`), Kolmogorov-Smirnov (`ks`), Cramer-vonMises (`cvm`) and Pearson (`ps`). The default is Shapiro-Francia (`"sf"`).
`digits`	a integer value indicating the numer of decimal places.
`pvalue.plot`	a logical value indicating if the p-value should be presented in the Quantile-Quantile plot.

Details

The function is a wrapper of ad.test, sf.test, lillie.test, cvm.test, and pearson.test.

Value

`tab` a table of p-values for each of the five normality tests.

`plot` a Quantile-Quantile plot.

Examples

```
library(magrittr)
data(iris)

iris %>% select(-Species) %>% nt_norm_test()
iris %>% nt_norm_test(group = Species)
```

`nt_simple_cox` *Simple Cox Regression*

Description

Performing simple Cox regression.

Usage

```
nt_simple_cox(data, time, status, ..., cluster = FALSE, strata = NULL,
  format = TRUE, digits = 2, digits.p = 3, save = FALSE,
  file = "simple_cox")
```

Arguments

`data`	a data frame with the variables.
`time`	a numeric vector with the follow-up time.
`status`	a numeric vector indicating status, 0 = censored, 1 = event at time.
`...`	character values indicating confounding variables.
`cluster`	a character vector containing the cluster variable.
`strata`	a character vector containing the strata variable.
`format`	a logical value indicating whether the output should be formatted.
`digits`	a numerical value defining of digits to present the results.
`digits.p`	a numerical value defining number of digits to present the p-values.
`save`	a logical value indicating whether the output should be saved as a csv file.
`file`	a character indicating the name of output file in csv format to be saved.

Examples

```
library(survival)
library(dplyr)
library(magrittr)
data(ovarian)

ovarian_nt <- ovarian %>% mutate(resid.ds = ql_var(resid.ds,
                                              from = 1:2,
                                              to = c("no", "yes"),
                                              label = "Residual Disease"),
                             ecog.ps = ql_var(ecog.ps,
                                             from = 1:2,
                                             to = c("I", "II"),
                                             label = "ECOG-PS"),
                             rx = ql_var(rx,
                                        from = 1:2,
                                        to = c("t1", "t2"),
                                        label = "Treatment"),
                             age = qt_var(age,
                                         label = "Age"))
ovarian_nt %>% nt_simple_cox(time = futime, status = fustat)
```

`nt_simple_logistic` *Simple Logistic Regression*

Description

Performing simple Logistic regression.

Usage

```
nt_simple_logistic(data, response, ..., format = TRUE, digits = 2,
  digits.p = 3, save = FALSE, file = "simple_logistic")
```

Arguments

`data`	a data frame with the variables.
`response`	a character value indicating the response variable.
`...`	character values indicating confounding variables.
`format`	a logical value indicating whether the output should be formatted.
`digits`	a numerical value defining of digits to present the results.
`digits.p`	a numerical value defining number of digits to present the p-values.
`save`	a logical value indicating whether the output should be saved as a csv file.
`file`	a character indicating the name of output file in csv format to be saved.

Examples

```
library(titanic)
library(magrittr)
library(dplyr)

data(titanic_train)
titanic_nt <- titanic_train %>%
 mutate(Sex = ql_var(Sex,
                     from = c("male", "female"),
                     to = c("Male", "Female")),
        Pclass = ql_var(Pclass,
                        from = 1:3,
                        to = c("I", "II", "III"),
                        label = "Passenger Class"),
        Embarked = ql_var(Embarked,
                          from = c("C", "Q", "S"),
                          to = c("Cherbourg", "Queenstown", "Southampton")))

titanic_nt %>% select(Survived, Sex, Age, Pclass, Embarked) %>%
 nt_simple_logistic(response = Survived, Age)
```

put_together *Put Together*

Description

Join a descriptive table and a p_value table.

Usage

```
put_together(descriptive.tab, test.tab, digits = 3, alpha = 0.05,
  save = FALSE, file = "table")
```

Arguments

`descriptive.tab`	a data frame of class "descriptive".
`test.tab`	a data frame of class "p_value".
`digits`	a numeric value specifying the number of digits for the p values.
`alpha`	a numeric value specifying the significance level to indicate statistically significant multiple comparisons.
`save`	a logical value indicating whether the output should be saved as a csv file.
`file`	a character value indicating the name of output file in csv format to be saved.

Value

A data frame similar to the `descriptive.tab` with an additional column of p values from the `p_value.tab`

Examples

```
library(dplyr)
library(magrittr)

iris_nt <- iris %>% filter(Species != "versicolor")
tab01 <- nt_describe(iris_nt, group = Species)
tab02 <- nt_compare_tg(iris_nt, group = Species)
tab <- put_together(tab01, tab02)
```

---

ql_var *Format qualitative variables*

Description

Recode factors, change the order of levels and add a label to the qualitative variable.

Usage

```
ql_var(var, from = NULL, to = NULL, order = NULL, label = NULL)
```

Arguments

`var`	vector to be formatted.
`from`	an optional vector of original levels that `var` might have taken. It also requires `to`.
`to`	an optional vector of values to recode the levels of `from`. It also requires `to`.
`order`	an optional vector of values to establish the level order using the same values of `from`.
`label`	an optional character to be attributed as a label to `var`.

Value

a vector of class "factor" recoded if `from` and `to` are provided, with an attribute label if `label` is provided, and reordered if `order` is provided.

Examples

```
library(dplyr)
library(magrittr)
data(iris)

iris_nt <- iris %>% mutate(Species = ql_var(Species,
from = c("setosa", "versicolor", "virginica"),
to = c("Setosa", "Versicolor", "Virginica"), label = "Species",
order = c("Virginica", "Setosa", "Versicolor")))

iris_nt %>% nt_describe(group = Species)
```

qt_var *Format qualitative variables*

Description

Recode factors, change the order of levels and add a label to the qualitative variable.

Usage

```
qt_var(var, label = NULL, unit = NULL)
```

Arguments

var	vector to be formatted.
label	an optional character to be attributed as a label to var.
unit	an optional character to be attributed as a unit to var.

Value

a vector with an attribute label if `label` is provided, and an attribute unit if `unit` is provided.

Examples

```
library(dplyr)
library(magrittr)
data(iris)

iris_nt <- iris %>%
mutate(Sepal.Length = qt_var(Sepal.Length, label = "Sepal Length", unit = "cm"),
      Sepal.Width = qt_var(Sepal.Width, label = "Sepal Width", unit = "cm"),
      Petal.Length = qt_var(Petal.Length, label = "Petal Length", unit = "cm"),
      Petal.Width = qt_var(Petal.Width, label = "Petal Length", unit = "cm"))
iris_nt %>% nt_describe(group = Species)
```

StatStepribbon *ntimes-ggproto*

Description

ntimes-ggproto

References

https://groups.google.com/forum/?fromgroups=#!topic/ggplot2/9cFWHaH1CPs `stat_stepribbon` *Step ribbon statistic*

Description

Provides stairstep values for ribbon plots

Usage

```
stat_stepribbon(mapping = NULL, data = NULL, geom = "ribbon",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, direction = "hv", ...)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame.`, and will be used as the layer data.
`geom`	which geom to use; defaults to "‘ribbon‘"
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`direction`	`hv` for horizontal-veritcal steps, ‘vh‘ for vertical-horizontal steps
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

References

https://groups.google.com/forum/?fromgroups=#!topic/ggplot2/9cFWHaH1CPs

`std_barplot` *Standard barplot*

Description

A function to plot a barplot without groups.

Usage

```
std_barplot(var, var.label, ylab)
```

Arguments

`var`	a character vector.
`var.label`	a character value specifying the variable label.
`ylab`	a character value specifying the y axis label.

Details

This function defines the standard barplot without groups to be plotted by the function `nt_barplot`. It can be modified by the user. See more details in `geom_bar`.

Value

a ggplot object.

`std_barplot_group` *Standard barplot by group*

Description

A function to plot a barplot with groups.

Usage

```
std_barplot_group(var, group, var.label, group.label, ylab)
```

Arguments

`var`	a character vector.
`group`	a character vector.
`var.label`	a character value specifying the variable label.
`group.label`	a character value specifying the group label.
`ylab`	a character value specifying the y axis label.

Details

This function defines the standard barplot with groups to be plotted by the function `nt_barplot`. It can be modified by the user. See more details in `geom_bar`.

Value

a ggplot object.

`std_boxplot` *Standard boxplot*

Description

A function to plot a boxplot without groups.

Usage

```
std_boxplot(var, var.label)
```

Arguments

`var`	a numeric vector.
`var.label`	a character value specifying the variable label.

Details

This function defines the standard boxplot without groups to be plotted by the function `nt_boxplot`. It can be modified by the user. See more details in `geom_boxplot`.

Value

a ggplot object.

`std_boxplot_group` *Standard boxplot by group*

Description

A function to plot a boxplot with groups.

Usage

```
std_boxplot_group(var, group, var.label, group.label)
```

Arguments

`var`	a numeric vector.
`group`	a character vector.
`var.label`	a character value specifying the variable label.
`group.label`	a character value specifying the group label.

Details

This function defines the standard boxplot with groups to be plotted by the function `nt_boxplot`. It can be modified by the user. See more details in `geom_boxplot`.

Value

a ggplot object.

---

std_dotplot *Standard dotplot*

Description

A function to plot a dotplot without groups.

Usage

```
std_dotplot(var, binwidth, var.label)
```

Arguments

`var`	a numeric vector.
`binwidth`	a numerical value specifying the bin width.
`var.label`	a character value specifying the variable label.

Details

This function defines the standard dotplot without groups to be plotted by the function `nt_dotplot`. It can be modified by the user. See more details in `geom_dotplot`.

Value

a ggplot object.

---

std_dotplot_group *Standard dotplot by group*

Description

A function to plot a dotplot with groups.

Usage

```
std_dotplot_group(var, group, binwidth, var.label, group.label)
```

Arguments

`var`	a numeric vector.
`group`	a character vector.
`binwidth`	a numerical value specifying the bin width.
`var.label`	a character value specifying the variable label.
`group.label`	a character value specifying the group label.

Details

This function defines the standard dotplot with groups to be plotted by the function `nt_dotplot`. It can be modified by the user. See more details in `geom_dotplot`.

Value

a ggplot object.

`std_km` *Standard Kaplan-Meier curve*

Description

A function to plot a Kaplan-Meier curve without groups.

Usage

```
std_km(time, status, xlab, ylab)
```

Arguments

`time`	a numeric vector.
`status`	a numeric vector of '0' and '1'.
`xlab`	a character value specifying the x axis label.
`ylab`	a character value specifying the y axis label.

Details

This function defines the standard of Kaplan-Meier curves without groups to be plotted by the function `nt_km`.

Value

a ggplot object.

`std_km_group` *Standard Kaplan-Meier curve by group*

Description

A function to plot a Kaplan-Meier curve with groups.

Usage

```
std_km_group(time, status, var, var.label, xlab, ylab)
```

Arguments

`time`	a numeric vector.
`status`	a numeric vector of '0' and '1'.
`var`	a character vector.
`var.label`	a character value specifying the group label.
`xlab`	a character value specifying the x axis label.
`ylab`	a character value specifying the y axis label.

Details

This function defines the standard of Kaplan-Meier curves with groups to be plotted by the function `nt_km`. It can be modified by the user.

Value

a ggplot object.

Index

∗Topic **datasets**
    StatStepribbon, 16

ad.test, *4*, *5*, *11*
aes(), *17*
aes_(), *17*

borders(), *17*

cvm.test, *4*, *5*, *11*

fortify(), *17*

geom_bar, *2*, *18*
geom_boxplot, *3*, *19*
geom_dotplot, *7*, *20*, *21*
ggplot(), *17*

layer(), *17*
leveneTest, *4*, *5*
lillie.test, *4*, *5*, *11*

nt_barplot, 2, *18*
nt_boxplot, 3, *19*
nt_compare_mg, 4
nt_compare_tg, 5
nt_describe, 6
nt_dotplot, 7, *20*, *21*
nt_km, 8, *21*, *22*
nt_multiple_cox, 9
nt_multiple_logistic, 10
nt_norm_test, 11
nt_simple_cox, 12
nt_simple_logistic, 13

pearson.test, *4*, *5*, *11*
put_together, 14

ql_var, 15
qt_var, 16

sf.test, *4*, *5*, *11*
stat_stepribbon, 17
StatStepribbon, 16
std_barplot, *2*, *8*, 18
std_barplot_group, *2*, *8*, 18
std_boxplot, *3*, 19
std_boxplot_group, *3*, 19
std_dotplot, *7*, 20
std_dotplot_group, *7*, 20
std_km, *8*, 21
std_km_group, *8*, 21