Package ‘visStatistics’

May 13, 2025

Type Package

Title Automated Selection and Visualisation of Statistical Hypothesis Tests

Version 0.1.3

Maintainer Sabine Schilling <sabineschilling@gmx.ch>

Description Automatically selects and visualises appropriate statistical hypothesis tests between a response and a feature variable in a data frame. The choice of test depends on the class, distribution, and sample size of the input variables, as well as the user-defined 'conf.level'. Well suited for web-based or server-side R applications.
Implemented tests: t.test(), wilcox.test(), aov(), oneway.test(), kruskal.test(), lm(), fisher.test(), chisq.test().
Tests for normality: shapiro.test(), ad.test(). Tests for equal variances: bartlett.test(). Post-hoc tests: TukeyHSD(), pairwise.wilcox.test().

License MIT + file LICENSE

URL https://github.com/shhschilling/visStatistics, https://shhschilling.github.io/visStatistics/

BugReports https://github.com/shhschilling/visStatistics/issues

Imports Cairo, graphics, grDevices, grid, multcompView, nortest, stats, utils, vcd

Suggests knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.2

Author Sabine Schilling [cre, aut, cph] (ORCID: <https://orcid.org/0000-0002-8318-9421>), Peter Kauf [ctb]

Repository CRAN

Date/Publication 2025-05-13 00:20:02 UTC

Contents

`colorscheme` `colorscheme(x)` *selects color scheme of graphical output. Function parameter NULL lists all available color schemes, 1 a color tuple of green and blue 2 a color tuple of dark green and turquoi, 3 a colorplaette as defined by RcolorBrewer*

Description

`colorscheme(x)` selects color scheme of graphical output. Function parameter NULL lists all available color schemes, 1 a color tuple of green and blue 2 a color tuple of dark green and turquoi, 3 a colorplaette as defined by RcolorBrewer

Usage

```
colorscheme(colorcode = NULL)
```

Arguments

`colorcode`	selects color scheme. parameters NULL: list of all available color schemes, 1: colortuple, 2, colortuple2, 3, ColorPalette

Value

selected color scheme, colors are given with their Hex Code #RRGGBB names

---

`counts_to_cases` *Convert data frame of counts to data frame of cases. data frame must contain a column with frequencies (counts) as generated by as.data.frame from a contingency table*

Description

Convert data frame of counts to data frame of cases. data frame must contain a column with frequencies (counts) as generated by as.data.frame from a contingency table

Usage

```
counts_to_cases(x, countcol = "Freq")
```

Arguments

`x`	a `data.frame` of counts generated from a contingency table.
`countcol`	character string, name of the column of x containing the counts. Default name of the column is 'Freq'.

Value

data frame of cases of dimension (total number of counts as sum of 'Freq' in x) times 2.

Examples

```
counts_to_cases(as.data.frame(HairEyeColor[, , 1]), countcol = "Freq")
```

`get_samples_fact_inputfile`
Selects columns defined by characters varsample and varfactor from a data.frame

---

Description

Selects columns defined by characters `varsample` and `varfactor` from `dataframe`, returns selected columns with their names.

Usage

```
get_samples_fact_inputfile(dataframe, varsample, varfactor)
```

Arguments

`dataframe`	`data.frame` or `list` containing at least two columns with column headings of data type `character`.Data must be column wise ordered.
`varsample`	column name of dependent variable in dataframe, datatype `character`
`varfactor`	column name of independent variable in dataframe, datatype `character`

Value

selected columns, `sample`, `factor`, `name_of_sample` (character string equaling varsample), `name_of_factor` (character string equaling varsample)

Examples

```
get_samples_fact_inputfile(trees, "Girth", "Height")
```

`openGraphCairo` *Cairo wrapper function*

Description

Cairo wrapper function returning NULL if not `type` is specified

Usage

```
openGraphCairo(
  width = 640,
  height = 480,
  fileName = NULL,
  type = NULL,
  fileDirectory = getwd(),
  pointsize = 12,
  bg = "transparent",
  canvas = "white",
  units = "px",
  dpi = 150
)
```

Arguments

`width`	see `Cairo()`
`height`	see `Cairo()`
`fileName`	name of file to be created. Does not include both file extension '.`type`' and file `filedirectory`. Default file name 'visstat_plot'.
`type`	Supported output types are 'png', 'jpeg', 'pdf', 'svg', 'ps' and 'tiff'. See `Cairo()`

`fileDirectory`	path of directory, where plot is stored. Default current working directory.
`pointsize`	see `Cairo()`
`bg`	see `Cairo()`
`canvas`	see `Cairo()`
`units`	see `Cairo()`
`dpi`	DPI used for the conversion of units to pixels. Default value 150.

Details

`openGraphCairo()` `Cairo()` wrapper function. Differences to `Cairo`: a) prematurely ends the function call to `Cairo()` returning NULL, if no output `type` of types 'png', 'jpeg', 'pdf', 'svg', 'ps' or 'tiff' is provided. b) The `file` argument of the underlying Cairo function is generated by `file.path(fileDirectory,paste(fileName,'.', type, sep = ''))`.

Value

NULL, if no `type` is specified. Otherwise see `Cairo()`

Examples

```
##  adapted from example in \code{Cairo()}
openGraphCairo(fileName = "normal_dist", type = "pdf", fileDirectory = tempdir())
plot(rnorm(4000), rnorm(4000), col = "#ff000018", pch = 19, cex = 2)
dev.off() # creates a file 'normal_dist.pdf' in the directory specified in fileDirectory
# ## remove the plot from fileDirectory
file.remove(file.path(tempdir(), "normal_dist.pdf"))
```

`saveGraphVisstat` *Saves Graphical Output*

Description

Closes all graphical devices with `dev.off()` and saves the output only if both `fileName` and `type` are provided.

Usage

```
saveGraphVisstat(
  fileName = NULL,
  type = NULL,
  fileDirectory = getwd(),
  oldfile = NULL
)
```

Arguments

`fileName`	name of file to be created in directory `fileDirectory` without file extension '`.type`'.
`type`	see `Cairo()`.
`fileDirectory`	path of directory, where graphic is stored. Default setting current working directory.
`oldfile`	old file of same name to be overwritten

Value

NULL, if no `type` or `fileName` is provided, TRUE if graph is created

Examples

```
# very simple KDE (adapted from example in Cairo())
openGraphCairo(type = "png", fileDirectory = tempdir())
plot(rnorm(4000), rnorm(4000), col = "#ff000018", pch = 19, cex = 2)
# save file 'norm.png' in directory specified in fileDirectory
saveGraphVisstat("norm", type = "png", fileDirectory = tempdir())
file.remove(file.path(tempdir(), "norm.png")) # remove file 'norm.png'
```

`visstat` *Automated Visualization of Statistical Hypothesis Testing*

Description

`visstat()` provides automated visualization and selection of a statistical hypothesis test between a response and a feature variable in a given `data.frame` named `dataframe`, selecting a test that is appropriate under the data's type, distribution, sample size, and the specified `conf.level`. The data in `dataframe` must be structured column-wise, where `varsample` and `varfactor` are `character` strings corresponding to the column names of the response and feature variables, respectively. The automatically generated output figures illustrate the selected statistical hypothesis test, display the main test statistics, and include assumption checks and post hoc comparisons when applicable. The primary test results are returned as a list object.

Usage

```
visstat(
  dataframe,
  varsample,
  varfactor,
  conf.level = 0.95,
  numbers = TRUE,
  minpercent = 0.05,
  graphicsoutput = NULL,
  plotName = NULL,
  plotDirectory = getwd()
)
```

Arguments

`dataframe`	`data.frame` containing at least two columns. Data must be column wise ordered.
`varsample`	column name of the dependent variable (response) in `dataframe`, datatype `character`. `varsample` must be one entry of the list `names(dataframe)`.
`varfactor`	column name of the independent variable (feature) in `dataframe`, datatype `character.varsample` must be one entry of the list `names(dataframe)`.
`conf.level`	confidence level
`numbers`	a logical indicating whether to show numbers in mosaic count plots.
`minpercent`	number between 0 and 1 indicating minimal fraction of total count data of a category to be displayed in mosaic count plots.
`graphicsoutput`	saves plot(s) of type "png", "jpg", "tiff" or "bmp" in directory specified in `plotDirectory`. If graphicsoutput=NULL, no plots are saved.
`plotName`	graphical output is stored following the naming convention "plotName.graphicsoutput" in `plotDirectory`. Without specifying this parameter, plotName is automatically generated following the convention "statisticalTestName_varsample_varfactor".
`plotDirectory`	specifies directory, where generated plots are stored. Default is current working directory.

Details

Decision logic (for more details, please refer to the package's `vignette`).

Throughout, data of class `numeric` or `integer` are referred to as numerical, while data of class `factor` are referred to as categorical. The significance level α is defined as one minus the confidence level, given by the argument `conf.level`.' Assumptions of normality and homoscedasticity are considered met when the corresponding test yields a p-value greater than alpha = 1 - 'conf.level'.

The choice of statistical tests performed by the function `visstat()` depends on whether the data are numerical or categorical, the number of levels in the categorical variable, the distribution of the data, and the chosen `conf.level()`.

The function prioritizes interpretable visual output and tests that remain valid under their assumptions, following the decision logic below:

(1) When the response is numerical and the predictor is categorical, tests of central tendency are performed. If the categorical predictor has two levels: - Welch's t-test (`t.test()`) is used if both groups have more than 30 observations (Lumley et al. (2002) <doi:10.1146/annurev.publheath.23.100901.140546>). - For smaller samples, normality is assessed using `shapiro.test()`. If both groups return p-values greater than α, Welch's t-test is applied; otherwise, the Wilcoxon rank-sum test (`wilcox.test()`) is used.

For predictors with more than two levels: - An ANOVA model (`aov()`) is initially fitted. - Residual normality is tested with `shapiro.test()` and `ad.test()`. If $p > \alpha$ for either test, normality is assumed. - Homogeneity of variance is tested with `bartlett.test()`: - If $p > \alpha$, use ANOVA with TukeyHSD(). - If $p \leq \alpha$, use oneway.test() with TukeyHSD(). - If residuals are not normal, use kruskal.test() with pairwise.wilcox.test().

(2) When both the response and predictor are numerical, a linear model (lm()) is fitted, with residual diagnostics and a confidence band plot.

(3) When both variables are categorical, visstat() uses chisq.test() or fisher.test() depending on expected counts, following Cochran's rule (Cochran (1954) <doi:10.2307/3001666>).

Implemented main tests: t.test(), wilcox.test(), aov(), oneway.test(), lm(), kruskal.test(), fisher.test(), chisq.test().

Implemented tests for assumptions:

- Normality: shapiro.test() and ad.test().
- Heteroscedasticity: bartlett.test().

Implemented post hoc tests:

- TukeyHSD() for aov() and oneway.test().
- pairwise.wilcox.test() for kruskal.test().

Value

list containing statistics of automatically selected test meeting assumptions. All values are returned as invisible copies. Values can be accessed by assigning a return value to visstat.

See Also

https://shhschilling.github.io/visStatistics/

Examples

```
## Welch Two Sample t-test (calling t.test())
visstat(mtcars, "mpg", "am")

## Wilcoxon rank sum test (calling wilcox.test())
grades_gender <- data.frame(
  Sex = as.factor(c(rep("Girl", 20), rep("Boy", 20))),
  Grade = c(
    19.3, 18.1, 15.2, 18.3, 7.9, 6.2, 19.4,
    20.3, 9.3, 11.3, 18.2, 17.5, 10.2, 20.1, 13.3, 17.2, 15.1, 16.2, 17.3,
    16.5, 5.1, 15.3, 17.1, 14.8, 15.4, 14.4, 7.5, 15.5, 6.0, 17.4,
    7.3, 14.3, 13.5, 8.0, 19.5, 13.4, 17.9, 17.7, 16.4, 15.6
  )
)
visstat(grades_gender, "Grade", "Sex")

## One-way analysis of means (oneway.test())
anova_npk <- visstat(npk, "yield", "block")
anova_npk # prints summary of tests

## Kruskal-Wallis rank sum test (calling kruskal.test())
visstat(iris, "Petal.Width", "Species")
visstat(InsectSprays, "count", "spray")

## Linear regression
visstat(trees, "Girth", "Height", conf.level = 0.99)

## Pearson's Chi-squared test and mosaic plot with Pearson residuals
### Transform array to data.frame
HairEyeColorDataFrame <- counts_to_cases(as.data.frame(HairEyeColor))
visstat(HairEyeColorDataFrame, "Hair", "Eye")

## 2x2 contingency tables with Fisher's exact test and mosaic plot
## with Pearson residuals
HairEyeColorMaleFisher <- HairEyeColor[, , 1]
### slicing out a 2 x2 contingency table
blackBrownHazelGreen <- HairEyeColorMaleFisher[1:2, 3:4]
blackBrownHazelGreen <- counts_to_cases(as.data.frame(blackBrownHazelGreen))
fisher_stats <- visstat(blackBrownHazelGreen, "Hair", "Eye")
fisher_stats # print out summary statistics



## Saving the graphical output in directory plotDirectory
## A) saving graphical output of type "png" in temporary directory tempdir()
##    with default naming convention:
visstat(blackBrownHazelGreen, "Hair", "Eye",
  graphicsoutput = "png",
  plotDirectory = tempdir()
)

## remove graphical output from plotDirectory
file.remove(file.path(tempdir(), "chi_squared_or_fisher_Hair_Eye.png"))
file.remove(file.path(tempdir(), "mosaic_complete_Hair_Eye.png"))

## B) Specifying pdf as output type:
visstat(iris, "Petal.Width", "Species",
  graphicsoutput = "pdf",
  plotDirectory = tempdir()
)

## remove graphical output from plotDirectory
file.remove(file.path(tempdir(), "kruskal_Petal_Width_Species.pdf"))

## C) Specifiying plotName overwrites default naming convention
visstat(iris, "Petal.Width", "Species",
  graphicsoutput = "pdf",
  plotName = "kruskal_iris", plotDirectory = tempdir()
)
## remove graphical output from plotDirectory
file.remove(file.path(tempdir(), "kruskal_iris.pdf"))
```

`vis_anova_assumptions` *Visualisation of the normality distribution of the standardised residuals of the ANOVA*

Description

`vis_anova_assumptions` checks for normality of the standardised residuals of the ANOVA. Both the Shapiro-Wilk test `shapiro.test()` and the Anderson-Darling test `ad.test()` check the null that the standardised residuals are normally distributed. It generates a scatter plot of the standardised residuals versus the fitted mean values of the linear models for each level of `fact`. Furthermore a normal QQ plot of the standardised residuals is generated. The null of homogeneity of variances of each factor level is tested with the `bartlett.test()`.

Usage

```
vis_anova_assumptions(
  samples,
  fact,
  conf.level = 0.95,
  samplename = "",
  factorname = "",
  cex = 1
)
```

Arguments

`samples`	vector containing dependent variable, datatype numeric
`fact`	vector containing independent variable, datatype factor
`conf.level`	confidence level, 0.95=default
`samplename`	name of sample used in graphical output, dataype character , ”=default
`factorname`	name of sample used in graphical output, dataype character, ”=default
`cex`	number indicating the amount by which plotting text and symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.

Value

`list` containing the test statistics of the anova, the p values generated by the Shapiro-Wilk test `shapiro.test()`, the Anderson-Darling test `ad.test()` and the `bartlett.test()`.

Examples

```
ToothGrowth$dose <- as.factor(ToothGrowth$dose)
vis_anova_assumptions(ToothGrowth$len, ToothGrowth$dose)

vis_anova_assumptions(ToothGrowth$len, ToothGrowth$supp)
vis_anova_assumptions(iris$Petal.Width, iris$Species)
```

Index