Package 'visStatistics'

May 8, 2026

Type Package

Title Automated Selection and Visualisation of Statistical Hypothesis Tests

Version 0.1.7

Maintainer Sabine Schilling <sabineschilling@gmx.ch>

Description Automatically selects
and visualises statistical hypothesis tests between two vectors,
based on their class, distribution, sample size, and a user-defined
confidence level (conf.level). Visual outputs - including box plots, bar charts,
regression lines with confidence bands, mosaic plots,
residual plots, and Q-Q plots - are annotated with relevant test statistics,
assumption checks, and post-hoc analyses where applicable.
The algorithmic workflow helps the user focus on the interpretation of test
results rather than test selection. It is particularly suited for quick data
analysis, e.g., in statistical consulting projects or educational settings.
The test selection algorithm proceeds as follows:
Input vectors of class numeric or integer are
considered numerical; those of class factor are considered categorical.
Assumptions of residual normality and homogeneity of variances are
considered met if the corresponding test yields a p-value greater than the
significance level alpha = 1 - conf.level.
(1) When the response vector is numerical and the
predictor vector is categorical, a test of central tendencies is selected.
If the categorical predictor has exactly two levels,
t.test() is applied when group sizes exceed 30
(Lumley et al. (2002) <doi:10.1146/annurev.publhealth.23.100901.140546>).
For smaller samples, normality of residuals is tested using shapiro.test();
if met, t.test() is used; otherwise, wilcox.test().
If the predictor is categorical with more than two levels, an
aov() is initially fitted. Residual normality is evaluated using both
shapiro.test() and ad.test(); residuals are considered approximately normal
if at least one test yields a p-value above alpha. If this assumption is met,
bartlett.test() assesses variance homogeneity.
If variances are homogeneous, aov() is used; otherwise oneway.test().
Both tests are followed by TukeyHSD().

If residual normality cannot be assumed, kruskal.test() is followed by pairwise.wilcox.test().
(2) When both the response and predictor vectors are numerical, a simple linear regression model is fitted using lm().
(3) When both vectors are categorical, Cochran's rule (Cochran (1954) <doi:10.2307/3001666>) is applied to test independence either by chisq.test() or fisher.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

BuildVignettes true

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.2

Author Sabine Schilling [cre, aut, cph] (ORCID: <https://orcid.org/0000-0002-8318-9421>, year: 2025), Peter Kauf [ctb]

Repository CRAN

Date/Publication 2025-05-28 19:50:02 UTC

Contents

`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")
```

`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"))
```

`plot.visstat` *Report only saved plot files for visstat objects*

Description

This method reports the file paths of plots that were saved to disk during the statistical analysis, if the user requested file output (e.g., PNG or PDF) via the `graphicsoutput` argument.

Usage

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

Arguments

`x`	An object of class `"visstat"`, returned by `visstat()`.
`...`	Currently unused. Included for S3 method compatibility.

Details

Note that plots are shown during execution of `visstat()`, even if no files are saved. This method does not re-render or replay those plots.

All file-based plots in `visStatistics` are generated using the `Cairo()` device and their file paths are collected in the object's `"plot_paths"` attribute.

In interactive sessions, this method prints the list of saved files. In non-interactive contexts, it suppresses output.

The `visstat()` function may produce several plots per test type (e.g., main effect, post hoc comparisons, assumption checks). These are saved to disk, and their file paths are collected in the object's `"plot_paths"` attribute.

In an interactive session, `plot()` will print these paths to the console. In non-interactive sessions, it quietly lists them. No plots are shown or rendered within R.

Value

Invisibly returns `x`. Used for its side effect of reporting file paths.

See Also

`visstat`, `Cairo`, `print.visstat`, `summary.visstat`

print.visstat *Print method for visstat objects*

Description

Displays a brief summary of the statistical test results and, if available, assumption tests and post hoc comparisons.

Usage

```
## S3 method for class 'visstat'
print(x, ...)
```

Arguments

x	An object of class "visstat", returned by visstat().
...	Currently unused. Included for S3 method compatibility.

Details

This function is automatically called when a visstat object is printed to the console. It provides a quick overview of the statistical analysis results.

Value

The object x, invisibly.

See Also

summary.visstat, plot.visstat, visstat

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'
```

---

summary.visstat *Summary method for visstat objects*

Description

Displays the full statistical test results and, if available, assumption tests and post hoc comparisons.

Usage

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

Arguments

object	An object of class "visstat", returned by visstat().
...	Currently unused. Included for S3 method compatibility.

Details

This method provides a full textual report of the statistical test results returned by visstat(), and prints the contents of posthoc_summary if present.

Value

The object object, invisibly.

See Also

print.visstat, visstat

---

visstat *Wrapper for visstat_core allowing two different input styles*

Description

A wrapper around the core function visstat_core defining the decision logic for statistical hypothesis testing and visualisation between two variables of class "numeric", "integer", or "factor".

Usage

```
visstat(
  x,
  y,
  ...,
  conf.level = 0.95,
  numbers = TRUE,
  minpercent = 0.05,
  graphicsoutput = NULL,
  plotName = NULL,
  plotDirectory = getwd()
)
```

Arguments

x	A vector of class "numeric", "integer", or "factor" (standardised usage), or a data.frame containing the relevant columns (backward-compatible usage).
y	A second vector (standardised usage), or a character string specifying the name of a column in x (backward-compatible usage).
...	If x is a data frame and y is a character string, an additional character string must follow, naming the second column.
conf.level	Confidence level for statistical inference; default is 0.95.
numbers	Logical. Whether to annotate plots with numeric values.
minpercent	Minimum proportion (between 0 and 1) required to display a category in plots.
graphicsoutput	Optional. Output format for plots (e.g., "pdf", "png").
plotName	Optional. File name prefix for saving plot output.
plotDirectory	Directory in which to save plots; defaults to the current working directory.

Details

This wrapper supports two input formats:

- **Standardised form:** `visstat(x, y)`, where both `x` and `y` are vectors of class `"numeric"`, `"integer"`, or `"factor"`.
- **Backward-compatible form:** `visstat(dataframe, "name_of_y", "name_of_x")`, where both character strings refer to column names in `dataframe`. This is equivalent to: `visstat(dataframe[["name_of_x"]], dataframe[["name_of_y"]])`.

The interpretation of `x` and `y` depends on the variable classes: In the following, data of class `numeric` or `integer` are both referred to by their common mode `numeric`

- If one variable is numeric and the other a factor, the numeric vector must be passed as `y` and the factor as `x`. This supports tests of central tendencies (e.g., t-test, ANOVA, Wilcoxon).
- If both variables are numeric, a linear model is fitted with `y` as the response and `x` as the predictor.
- If both variables are factors, an association test (Chi-squared or Fisher's exact) is used. The test result is invariant to variable order, but visualisations (e.g., axis layout, bar orientation) depend on the roles of `x` and `y`.

This wrapper standardises the input and calls `visstat_core`, which selects and executes the appropriate test with visual output and assumption diagnostics.

Value

A list as returned by `visstat_core`, containing statistical results and graphical outputs.

See Also

the core function `visstat_core`, the package's vignette `vignette("visStatistics")` for the overview, and the accompanying webpage https://shhschilling.github.io/visStatistics/.

Examples

```
## Standardised usage (preferred):
visstat(mtcars$am, mtcars$mpg)

## Backward-compatible usage (same result):
visstat(mtcars, "mpg", "am")

## Wilcoxon rank sum 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$Sex, grades_gender$Grade)

## Welch's one-way ANOVA
visstat(npk$block, npk$yield)

## Kruskal-Wallis
visstat(iris$Species, iris$Petal.Width)

## Simple linear regression
visstat(trees$Height, trees$Girth, conf.level = 0.99)

## Chi-squared
HairEyeColorDataFrame <- counts_to_cases(as.data.frame(HairEyeColor))
visstat(HairEyeColorDataFrame$Eye, HairEyeColorDataFrame$Hair)

## Fisher's test
HairEyeColorMaleFisher <- HairEyeColor[, , 1]
blackBrownHazelGreen <- HairEyeColorMaleFisher[1:2, 3:4]
blackBrownHazelGreen <- counts_to_cases(as.data.frame(blackBrownHazelGreen))
visstat(blackBrownHazelGreen$Eye, blackBrownHazelGreen$Hair)

## Save PNG
visstat(blackBrownHazelGreen$Hair, blackBrownHazelGreen$Eye,
        graphicsoutput = "png", plotDirectory = tempdir())

## Save PDF
visstat(iris$Species, iris$Petal.Width, graphicsoutput = "pdf",
          plotDirectory = tempdir())

## Custom plot name
visstat(iris$Species, iris$Petal.Width,
        graphicsoutput = "pdf", plotName = "kruskal_iris", plotDirectory = tempdir())
```

`visstat_core` *Automated Visualization of Statistical Hypothesis Testing*

Description

`visstat_core()` provides automated selection and visualization of a statistical hypothesis test between a two vectors in a given `data.frame` named `dataframe` based on the data's type, distribution, sample size, and the specified `conf.level`. `varsample` and `varfactor` are `character` strings corresponding to the column names of the chosen vectors in `dataframe`. These vectors must be of type `integer`, `numeric` or `factor`. 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_core(
```

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

Arguments

dataframe	data.frame with at least two columns.
varsample	character string matching a column name in dataframe. Interpreted as the response if the referenced column is of class numeric or integer and the column named by varfactor is of class factor.
varfactor	character string matching a column name in dataframe. Interpreted as the grouping variable if the referenced column is of class factor and the column named by varsample is of class numeric or integer.
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

The decision logic for selecting a statistical test is described below. For more details, please refer to the package's vignette("visstat_coreistics"). Throughout, data of class numeric or integer are referred to as numeric, while data of class factor are referred to as categorical. The significance level alpha 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 visstat_core() depends on whether the data are numeric or categorical, the number of levels in the categorical variable, the distribution of the data, and the chosen conf.level. The function prioritises interpretable visual output and tests that remain valid under their assumptions, following the logic below:

(1) When the response is numerical and the predictor is categorical, tests of central tendency are performed. If the predictor has two levels: t.test() is used if both groups have more than 30

observations (Lumley et al. (2002) <doi:10.1146/annurev.publhealth.23.100901.140546>). For smaller samples, normality is assessed using `shapiro.test()`. If both groups return p-values greater than `alpha`, `t.test()` is applied; otherwise, `wilcox.test()` is used. For predictors with more than two levels, `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`, `aov()` with `TukeyHSD()` is used. If `p <= alpha`, `oneway.test()` is applied with `TukeyHSD()`. If residuals are not normal, `kruskal.test()` with `pairwise.wilcox.test()` is used.

(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_core()` 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_core`.

See Also

See also the package's vignette `vignette("visStatistics")` for the overview, and the accompanying webpage https://shhschilling.github.io/visStatistics/.

Examples

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

## Wilcoxon rank sum test (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_core(grades_gender, "Grade", "Sex")

## Welch's oneway ANOVA not assuming equal variances (oneway.test())
anova_npk <- visstat_core(npk, "yield", "block")
anova_npk # prints summary of tests

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

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

## Pearson's Chi-squared test (chisq.test())
### Transform array to data.frame
HairEyeColorDataFrame <- counts_to_cases(as.data.frame(HairEyeColor))
visstat_core(HairEyeColorDataFrame, "Hair", "Eye")

## Fisher's exact test (fisher.test())
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_core(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_core(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_core(iris, "Petal.Width", "Species",
  graphicsoutput = "pdf",
  plotDirectory = tempdir()
)

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

## C) Specifying "plotName" overwrites default naming convention
visstat_core(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, datatype character , "=default
`factorname`	name of sample used in graphical output, datatypecharacter, "=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