

Package ‘visStatistics’

July 27, 2026

Type Package

Title Automated Selection and Visualisation of Statistical Hypothesis Tests

Version 0.3.0

Author Sabine Schilling [cre, aut, cph] (ORCID:
<<https://orcid.org/0000-0002-8318-9421>>, year: 2026),
Peter Kauf [ctb]

Maintainer Sabine Schilling <sabineschilling@gmx.ch>

Description Automated test selection, visualised. 'visStatistics' automatically selects and visualises statistical hypothesis tests comparing two vectors, based on their class and distribution. 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 shifts attention from ad-hoc test selection to visual diagnostic assessment and statistical interpretation. It is particularly suited for server-side R applications, where end users interact solely through a web interface to select data groups and receive a complete visual statistical analysis automatically. The same automation makes it useful in time-constrained contexts such as statistical consulting, where it reduces effort spent on test selection and leaves more room for interpretation. The implemented tests cover the most frequently applied inferential methods in biomedical research (Hayat et al. (2017) <[doi:10.1371/journal.pone.0179032](https://doi.org/10.1371/journal.pone.0179032)>). The test selection algorithm proceeds as follows: Input vectors of class numeric or integer are considered numerical; those of class factor are considered categorical; those of class ordered are considered ordinal. Assumptions of residual normality and homogeneity of variances are considered met if the corresponding test yields a p-value greater than the significance level $\alpha = 1 - \text{conf.level}$. (1) When the response is numerical and the predictor is categorical, a test comparing central tendencies is selected. In the default setting (`group_test = NULL`), residual normality is assessed at every group size using `shapiro.test()` applied to the standardised residuals of `lm()`. If

normality is not met, `wilcox.test()` is used when the predictor has two levels and `kruskal.test()` followed by `pairwise.wilcox.test()` otherwise. If normality is met, `levene.test()` assesses variance homogeneity. For two-level predictors, Student's `t.test(var.equal = TRUE)` is applied if variances are homogeneous and Welch's `t.test()` otherwise. For predictors with more than two levels, `aov()` followed by `TukeyHSD()` is applied if variances are homogeneous, and `oneway.test()` followed by `games.howell()` otherwise. Setting `group_test` to `"welch"` or `"rank"` bypasses these assumption tests and fixes the analysis to Welch-type or to rank-based tests, respectively.

(2) When both vectors are numerical, `lm()` is fitted by default (`correlation = FALSE`). If `correlation = TRUE`, Spearman rank correlation is performed. (3) When the response is ordinal, it is converted to numeric ranks and the non-parametric path from (1) is followed (Wilcoxon or Kruskal-Wallis). When both variables are ordinal and `correlation = TRUE`, Kendall's `tau_b` is used instead. (4) When both vectors are categorical, Cochran's rule (Cochran (1954) [doi:10.2307/3001666](https://doi.org/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, tools, utils, vcd

Suggests bookdown, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

BuildVignettes true

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

NeedsCompilation no

Repository CRAN

Date/Publication 2026-07-27 21:50:02 UTC

Contents

<code>bp.test</code>	3
<code>counts_to_cases</code>	4
<code>effect_size</code>	5
<code>games.howell</code>	8

<i>bp.test</i>	3
levene.test	9
openGraphCairo	10
plot.visstat	12
print.visstat	13
qq_lm_envelope	14
saveGraphVisstat	15
summary.visstat	16
visstat	17
visstat_core	21
vis_lm_assumptions	24
Index	26

<code>bp.test</code>	<i>Breusch-Pagan Test for Heteroscedasticity</i>
----------------------	--

Description

Performs the Breusch-Pagan test for heteroscedasticity in linear regression models. Tests the null hypothesis that the error variance is constant (homoscedasticity) against the alternative that the error variance depends on the fitted values.

Usage

`bp.test(model)`

Arguments

`model` A fitted linear model object (from `lm()`).

Details

Implements the Koenker variant of the Breusch-Pagan test, which regresses the squared raw residuals on the fitted values. The test statistic is:

$$BP = n \cdot R^2$$

where:

- n = sample size
- R^2 = coefficient of determination from auxiliary regression of e_i^2 on $\hat{y}_i$

Under the null hypothesis of homoscedasticity, the test statistic follows a chi-squared distribution: $BP \sim \chi^2(k - 1)$ where k is the number of parameters in the model (including intercept).

Large values of the test statistic (small p-values) provide evidence against homoscedasticity.

Value

An object of class "htest" with components:

statistic	the value of the chi-squared test statistic.
parameter	degrees of freedom.
p.value	the p-value of the test.
method	a character string indicating the test performed.
data.name	a character string giving the name of the model.

References

Breusch, T. S., and Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, 47(5), 1287-1294. DOI: 10.2307/1911963

Koenker, R. (1981). A note on studentizing a test for heteroscedasticity. *Journal of Econometrics*, 17(1), 107-112. DOI: 10.1016/0304-4076(81)90062-2

Examples

```
# Example with homoscedastic errors
set.seed(123)
x <- runif(100)
y <- 2 + 3 * x + rnorm(100, sd = 1)
model1 <- lm(y ~ x)
bp.test(model1) # Should not reject (p > 0.05)

# Example with heteroscedastic errors (variance increases with x)
set.seed(456)
x <- runif(100)
y <- 2 + 3 * x + rnorm(100, sd = 0.5 + 2 * x)
model2 <- lm(y ~ x)
bp.test(model2) # Should reject (p < 0.05)
```

counts_to_cases	<i>Convert data frame of counts to dataframe of cases. The data frame must contain a column with frequencies (counts) as generated by as.data.frame from a contingency table</i>
-----------------	--

Description

Convert data frame of counts to dataframe of cases. The 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

<code>x</code>	a <code>data.frame</code> of counts generated from a contingency table.
<code>countcol</code>	character string, name of the column of <code>x</code> 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")
```

effect_size	<i>Compute an effect-size estimate for a visstat result</i>
-------------	---

Description

`effect_size()` returns the effect-size estimate associated with a `visstat()` result. If `result$effect_size` is already present, it is returned unchanged. Otherwise, the estimate is computed from the test object stored in `result`; for some base R **stats** results, it is extracted directly from the returned object.

Usage

```
effect_size(result, x = NULL, y = NULL, ...)
```

Arguments

<code>result</code>	A list returned by <code>visstat()</code> or a compatible test result object; or, in raw-data mode, the first input vector <code>x</code> .
<code>x</code>	First input vector, matching the first argument of <code>visstat(x, y)</code> . Required when the effect size cannot be extracted from <code>result</code> alone. In raw-data mode this is the second input vector <code>y</code> .
<code>y</code>	Second input vector, matching the second argument of <code>visstat(x, y)</code> . Required when the effect size cannot be extracted from <code>result</code> alone.
<code>...</code>	Passed to <code>visstat()</code> in raw-data mode (e.g. <code>correlation</code> , <code>conf.level</code>).

Details

Notation used below: x and y are the two variables entering the selected analysis, N is the total number of non-missing observations, n_j is the sample size in group j , k is the number of groups, $\bar{y}_j$ is the mean of numeric vector y in group j , and s_j^2 is the variance of numeric vector y in group j .

The following estimates are computed internally:

- Student's two-sample `t.test(..., var.equal = TRUE)`: Hedges' $g_{s_p} = J(N - 2)(\bar{y}_1 - \bar{y}_2)/s_p$, where $s_p = \sqrt{((n_1 - 1)s_1^2 + (n_2 - 1)s_2^2)/(N - 2)}$ and $J(\nu) = \Gamma(\nu/2)/(\sqrt{\nu/2}\Gamma((\nu - 1)/2))$.
- Welch's two-sample `t.test(..., var.equal = FALSE)`: Hedges' $g_{s^*} = J(\nu^*)(\bar{y}_1 - \bar{y}_2)/s^*$, where $s^* = \sqrt{(s_1^2 + s_2^2)/2}$ and $\nu^* = ((n_1 - 1)(n_2 - 1)(s_1^2 + s_2^2)^2)/((n_2 - 1)s_1^4 + (n_1 - 1)s_2^4)$.
- Wilcoxon rank-sum test: signed rank-biserial correlation $r = 2W/(n_1 n_2) - 1$, where W is the statistic returned by `wilcox.test()` for the first group.
- Fisher's one-way ANOVA: omega-squared $\omega^2 = \nu_1(F - 1)/(\nu_1 F + \nu_2 + 1)$, where F is the ordinary one-way ANOVA statistic, $\nu_1 = k - 1$, and $\nu_2 = N - k$. Negative estimates are truncated to zero.
- Welch's one-way test: approximate omega-squared-type estimate $\nu_1(F_W - 1)/(\nu_1 F_W + \nu_2 + 1)$, where F_W is the Welch ANOVA statistic, $\nu_1 = k - 1$, and ν_2 is the usually fractional denominator degree of freedom returned by `oneway.test()`. Negative estimates are truncated to zero.
- Kruskal-Wallis test: Kelley-adjusted eta-squared based on H , $\eta_H^2 = (H - k + 1)/(N - k)$, where H is the Kruskal-Wallis statistic. Negative estimates are truncated to zero.
- Pearson's chi-squared test: Cramer's V for general $R \times C$ tables, $V = \sqrt{\chi^2/(N \cdot (\min(R, C) - 1))}$, where R and C are the numbers of rows and columns. For 2×2 tables this is phi, $\sqrt{\chi^2/N}$. The chi-squared statistic is used as supplied by `chisq.test()`.

The following estimates are extracted from existing result objects:

- R^2 from `summary(lm())$r.squared`, the coefficient of determination $R^2 = 1 - SS_{res}/SS_{tot}$, where SS_{res} is the residual and SS_{tot} the total sum of squares.
- Spearman's ρ from `cor.test(method = "spearman")$estimate`, the Pearson correlation of the ranks, $\rho = r(rank(x), rank(y))$.
- Kendall's τ_b from `cor.test(method = "kendall")$estimate`, $\tau_b = (n_c - n_d)/\sqrt{(n_0 - n_1)(n_0 - n_2)}$, where n_c and n_d are the numbers of concordant and discordant pairs, $n_0 = n(n - 1)/2$ the total number of pairs, and n_1 and n_2 the numbers of pairs tied in the response and in the predictor.
- The conditional maximum-likelihood odds ratio from `fisher.test()$estimate` and its confidence interval from `fisher.test()$conf.int` for 2×2 tables.

Value

A list with components `name`, `estimate`, `effect_size_method`, and optionally `conf.int`. In raw-data mode (`effect_size(x, y)`) the list additionally contains `selected_test`, the name of the test `visstat()` selected.

References

- Hedges, L. V. (1981). Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics*, 6(2), 107–128. doi:10.3102/10769986006002107.
- Delacre, M., Lakens, D., Ley, C., Liu, L., & Leys, C. (2021). Why Hedges' g 's based on the non-pooled standard deviation should be reported with Welch's t-test. *PsyArXiv*. doi:10.31234/osf.io/tu6mp.

Kerby, D. S. (2014). The simple difference formula: An approach to teaching nonparametric correlation. *Comprehensive Psychology*, 3, 11.IT.3.1. doi:10.2466/11.IT.3.1.

Albers, C., & Lakens, D. (2018). When power analyses based on pilot data are biased: Inaccurate effect size estimators and follow-up bias. *Journal of Experimental Social Psychology*, 74, 187–195. doi:10.1016/j.jesp.2017.09.004.

Kelley, T. L. (1935). An unbiased correlation ratio measure. *Proceedings of the National Academy of Sciences*, 21(9), 554–559. doi:10.1073/pnas.21.9.554.

Cohen, J. (2013). *Statistical power analysis for the behavioural sciences*. Routledge. doi:10.4324/9780203771587.

Examples

```
x <- ToothGrowth$supp
y <- ToothGrowth$len
tt <- list("t-test-statistics" = t.test(y ~ x, var.equal = TRUE))
effect_size(tt, x = x, y = y)

kw <- list(
  "Kruskal Wallis rank sum test" = kruskal.test(Petal.Width ~ Species,
    data = iris
  )
)
effect_size(kw, x = iris$Species, y = iris$Petal.Width)

tab <- matrix(c(10, 5, 4, 12), nrow = 2)
effect_size(chisq.test(tab))

## Raw-data mode: select the test with visstat() and return the effect
## size together with the name of the selected test.
effect_size(ToothGrowth$supp, ToothGrowth$len)

## Not run:
## Large-sample example with a statistically significant Student's
## t-test p-value but a small effect size, measured by Hedges' g
## using the pooled standard deviation. A small mean shift is added
## to noisy normal data. Because N is large, the t-test p-value
## becomes small, while Hedges' g remains close to zero.
## The residual Shapiro-Wilk p-value in the diagnostic panel is NA
## because shapiro.test() is limited to n <= 5000.
set.seed(20260525)
n <- 2501
mean_shift <- 0.1
group <- factor(rep(c("control", "treatment"), each = n))
response <- rnorm(2 * n) + rep(c(0, mean_shift), each = n)
res <- visstat(group, response)
res[["t-test-statistics"]]$method
res[["t-test-statistics"]]$p.value
res$effect_size$effect_size_method
res$effect_size$estimate

## End(Not run)
```

games.howell

Games-Howell Post-Hoc Test

Description

Performs pairwise comparisons using the package's own implementation of a Games-Howell-type unequal-variance post-hoc procedure. It does not assume equal variances or equal sample sizes and is used after Welch's ANOVA.

Usage

```
games.howell(samples, groups, conf.level = 0.95)
```

Arguments

<code>samples</code>	Numeric vector; the dependent variable.
<code>groups</code>	Factor or vector; the grouping variable.
<code>conf.level</code>	Numeric; confidence level for confidence intervals (default: 0.95).

Details

The implementation uses the Welch-Satterthwaite approximation for degrees of freedom and does not pool variances. P-values are adjusted using the Holm method to control family-wise error rate.

Value

A data frame with columns:

group1	First group in comparison
group2	Second group in comparison
mean_diff	Difference in means (group1 - group2)
se	Standard error of the difference
t	t-statistic
df	Degrees of freedom (Welch-Satterthwaite)
p_value	Unadjusted p-value
p_adj	Holm-adjusted p-value for multiple comparisons
ci_lower	Lower bound of confidence interval
ci_upper	Upper bound of confidence interval
significant	Logical; TRUE if $p_{adj} < (1 - \text{conf.level})$

Examples

```
# Convert dose to factor
ToothGrowth$dose <- as.factor(ToothGrowth$dose)

# Perform Games-Howell test
result <- games.howell(ToothGrowth$len, ToothGrowth$dose)
print(result)
```

levene.test	<i>Mean-centred Levene Test for Homogeneity of Variance</i>
-------------	---

Description

Performs Levene's original mean-centred test. It tests the null hypothesis that all groups have equal variances by testing whether the absolute deviations from group means are equal across groups. The function reproduces the behaviour of `leveneTest(y, g, center = mean, ...)` of the `car` package.

Usage

```
levene.test(y, g, data = NULL)
```

Arguments

<code>y</code>	A numeric response vector.
<code>g</code>	A grouping factor.
<code>data</code>	Optional data frame containing 'y' and 'g'.

Details

For each observation y_{ij} in group i , compute the absolute deviation from the group mean:

$$z_{ij} = |y_{ij} - \bar{y}_i|$$

where $\bar{y}_i$ is the mean of group i .

The test statistic is the F-statistic from a one-way ANOVA on the z_{ij} values:

$$F = \frac{(N - k) \sum_{i=1}^k n_i (\bar{z}_i - \bar{z})^2}{(k - 1) \sum_{i=1}^k \sum_{j=1}^{n_i} (z_{ij} - \bar{z}_i)^2}$$

where:

- k = number of groups
- N = total sample size
- n_i = sample size of group i

- $\bar{z}_i$ = mean of absolute deviations in group i
- $\bar{z}$ = overall mean of all absolute deviations

Under the null hypothesis of equal variances, the test statistic follows an F-distribution: $F \sim F(k - 1, N - k)$.

Value

An object of class "htest" with components:

statistic	the value of the F-statistic.
parameter	degrees of freedom: df1=k-1, df3=N-k, where k is the number of groups and N the total sample size
p.value	the p-value of the test.
method	a character string indicating the test performed.
data.name	a character string giving the name(s) of the data.

References

Levene, H. (1960). Robust tests for equality of variances. In I. Olkin (Ed.), Contributions to Probability and Statistics (pp. 278-292). Stanford University Press.

Examples

```
set.seed(123)
y <- c(rnorm(10), rnorm(10, sd = 2), rnorm(10, sd = 0.5))
g <- factor(rep(1:3, each = 10))
levene.test(y, g)

# Usage with data frame
df <- data.frame(response = y, group = g)
levene.test(response, group, data = df)

# Example with unequal variances (should reject null hypothesis)
set.seed(456)
y_unequal <- c(rnorm(15, sd = 1), rnorm(15, sd = 5), rnorm(15, sd = 0.2))
g_unequal <- factor(rep(c("A", "B", "C"), each = 15))
levene.test(y_unequal, g_unequal)
```

openGraphCairo

Cairo wrapper function with plot capture capability

Description

Cairo wrapper function returning NULL if not type is specified. Enhanced version that can capture plots for later replay.

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 = '.')). c) Can set up plot capture when capture_env is provided.

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	<i>Plot method for visstat objects</i>
--------------	--

Description

Replays captured plots or reports saved plot file paths from a visstat object.

Usage

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

Arguments

x	An object of class "visstat", returned by <code>visstat()</code> .
which	Integer selecting a single plot to display (1, 2, ...). If NULL (the default), all available plots are listed without being drawn.
...	Currently unused. Included for S3 method compatibility.

Details

When called without which, the method lists all available plots (either as file paths or as indices of captured plots). When called with which, the selected plot is displayed: for file-based output the stored path is printed, for captured plots the plot is replayed via `replayPlot()`.

Value

The object x, invisibly. Called for its side effect.

See Also

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

Examples

```
# File-based output: plot() lists stored paths
anova_path <- visstat(
  npk$block,
  npk$yield,
  graphicsoutput = "png",
  plotDirectory = tempdir()
)

plot(anova_path)

# Interactive output: plot() lists available plots,
# plot(obj, which = n) replays a specific one
linreg <- visstat(trees$Height, trees$Girth)
```

```
plot(linreg)
plot(linreg, which = 2)
```

print.visstat	<i>Print method for visstat objects</i>
---------------	---

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

Quick overview of the statistical analysis results.

Value

The object x, invisibly.

See Also

[summary.visstat](#), [plot.visstat](#), [visstat](#)

Examples

```
anova <- visstat(npk$block, npk$yield)
print(anova)
```

qq_lm_envelope

*Simulated Q-Q envelopes for linear model residuals***Description**

Computes pointwise and simultaneous Q-Q envelopes for internally studentised residuals from an unweighted `lm` or `aov` object by repeatedly simulating responses from the fitted normal-error model and refitting the model. The simultaneous envelope follows the Monte Carlo tolerance-band idea of Schuetzenmeister et al. (2012, doi:10.1080/03610918.2011.582560).

Usage

```
qq_lm_envelope(
  model,
  conf.level = 0.95,
  nsim = getOption("visStatistics.qq_nsim", 5000L),
  q.type = 2L,
  tol = 1e-04,
  max.iter = 100L
)
```

Arguments

<code>model</code>	An unweighted <code>lm</code> or <code>aov</code> object.
<code>conf.level</code>	Numeric confidence level for the envelopes.
<code>nsim</code>	Integer number of simulated refits.
<code>q.type</code>	Integer quantile type passed to <code>stats::quantile()</code> .
<code>tol</code>	Numeric tolerance for the simultaneous-band coverage search.
<code>max.iter</code>	Integer maximum number of bisection iterations.

Value

A list of class `qq_lm_envelope` with the observed sorted residuals, theoretical quantiles, pointwise bounds, simultaneous bounds, the simulated sorted residual matrix, and the achieved simultaneous coverage.

Examples

```
fit <- lm(mpg ~ wt, data = mtcars)
env <- qq_lm_envelope(fit, nsim = 100)
str(env)
```

saveGraphVisstat	<i>Saves Graphical Output with plot capture capability</i>
------------------	--

Description

Closes all graphical devices with `dev.off()` and saves the output only if both `fileName` and `type` are provided. Enhanced version that can capture plots before closing devices.

Usage

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

Arguments

<code>fileName</code>	name of file to be created in directory <code>fileDirectory</code> without file extension <code>'type'</code> .
<code>type</code>	see <code>Cairo()</code> .
<code>fileDirectory</code>	path of directory, where graphic is stored. Default setting current working directory.
<code>oldfile</code>	old file of same name to be overwritten
<code>capture_env</code>	Environment to store captured plots. If <code>NULL</code> , no capture occurs.

Value

`NULL`, if no `type` or `fileName` is provided or if no graphics file was created, file path 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

<code>object</code>	An object of class "visstat", returned by <code>visstat()</code> .
<code>...</code>	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](#)

Examples

```
anova <- visstat(npk$block, npk$yield)  
summary(anova)
```


visstat

*Wrapper for visstat_core Allowing Three Different Input Styles***Description**

`visstat()` is a wrapper around `visstat_core` that provides three input styles and dispatches to one of four analysis routes. Route 1 handles a numeric response with a categorical predictor. By default, Route 1 uses residual-based assumption diagnostics: the Shapiro–Wilk test gates mean-based versus rank-based analysis, and the Levene test then gates equal-variance versus Welch-type mean tests. Above 5000 observations, where `shapiro.test()` is undefined, the Anderson–Darling test takes over as the residual-normality gate. If the group sizes are unbalanced, the largest group standard deviations occur in the smallest groups, and the selected route is the equal-variance test or a rank-based test, the default route warns and points at `group_test = "welch"`, because those two routes can exceed the nominal significance level in that configuration. Alternatively, `group_test = "welch"` keeps Route 1 on the mean scale and forces Welch-type tests, while `group_test = "rank"` forces Wilcoxon/Kruskal–Wallis rank tests. Route 2 handles ordered responses with Wilcoxon/Kruskal–Wallis tests. Route 3 handles two numeric variables with linear regression by default, or Spearman correlation when `correlation = TRUE`. Route 4 handles two unordered factors with Pearson’s χ^2 test or Fisher’s exact test.

Usage

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

Arguments

- | | |
|---|---|
| x | For the formula interface: a formula of the form $y \sim x$, where y is the response variable and x is the predictor or grouping variable (requires data argument). For the standardised form: a vector of class "numeric", "integer", or "factor" representing the predictor or grouping variable. For the backward-compatible form: a <code>data.frame</code> containing the relevant columns. |
| y | For the formula interface: not used (variables are extracted from the formula). For the standardised form: a vector of class "numeric", "integer", or "factor" |

	representing the response variable. For the backward-compatible form: a character string specifying the name of the response variable column in <code>x</code> .
<code>...</code>	Named graphical arguments are forwarded to the plots selected by the analysis route for all three input forms. Only in the backward-compatible form, the first unnamed extra argument must specify the predictor or grouping column name in <code>x</code> . Ignored otherwise.
<code>data</code>	A <code>data.frame</code> containing the variables specified in the formula. Required when using the formula interface. Ignored for other input styles.
<code>conf.level</code>	Confidence level for statistical inference; default is 0.95.
<code>correlation</code>	Logical. If FALSE (default), performs simple linear regression analysis with confidence and prediction bands when both variables are numeric. If TRUE, performs Spearman correlation analysis with trend line only (no regression interpretation).
<code>numbers</code>	Logical. Whether to annotate plots with numeric values.
<code>minpercent</code>	Number between 0 and 1 indicating minimal fraction of total count data of a category to be displayed in mosaic count plots.
<code>group_test</code>	Optional character. For a numeric response and factor predictor, NULL keeps the default assumption gates, "welch" forces Welch-type mean tests, but still displays the assumption-diagnostic plot and warns when residual normality is rejected, whereas "rank" forces Wilcoxon/Kruskal-Wallis rank tests without assessing the assumptions.
<code>graphicsoutput</code>	Saves plot(s) of type "png", "jpeg", "pdf", "svg", "ps" or "tiff" in directory specified in <code>plotDirectory</code> . If NULL, no plots are saved. Any other value is not supported by <code>Cairo()</code> : it triggers a warning, no file is written, and no <code>plot_paths</code> attribute is returned.
<code>plotName</code>	Graphical output is stored following the naming convention " <code>plotName.graphicsoutput</code> " in <code>plotDirectory</code> . Without specifying this parameter, <code>plotName</code> is automatically generated following the convention " <code>statisticalTestName_varsample_varfactor</code> ".
<code>plotDirectory</code>	Specifies directory where generated plots are stored. Default is current working directory.

Details

This wrapper supports three input formats:

- (1) Formula interface: `visstat(y ~ x, data = df)`, where the formula specifies the response (`y`) and predictor (`x`) variables, and `data` is a data frame containing these variables.
- (2) Standardised form: `visstat(x, y)`, where both `x` and `y` are vectors of class "numeric", "integer", or "factor". Here `x` is the predictor or grouping variable and `y` is the response variable.
- (3) Backward-compatible form: `visstat(dataframe, "name_of_y", "name_of_x")`, where the first character string refers to the response variable and the second to the predictor or grouping variable. Both must be column names in `dataframe`. This form gives a warning and may be removed in a future version.

The interpretation of `x` and `y` depends on the variable classes. Throughout, data of class `numeric` or `integer` are referred to as `numeric`, while data of class `factor` are referred to as `categorical`:

If one variable is numeric and the other a factor, the numeric vector is the response (y) and the factor is the grouping variable (x). This is Route 1: residual normality and residual variance diagnostics select between Student/Welch mean tests and Wilcoxon/Kruskal–Wallis rank tests, unless `group_test` explicitly forces "welch" or "rank".

If both variables are numeric, a linear model is fitted with y as the response and x as the predictor, unless `correlation = TRUE`, which requests Spearman rank correlation.

If both variables are factors, unordered factors enter the Pearson- χ^2 /Fisher exact association route. Ordered responses with categorical predictors are analysed as rank-based group comparisons; if both variables are ordered and `correlation = TRUE`, Kendall's τ_b is used.

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

The Q-Q envelopes in the assumption diagnostics are simulated (see `qq_lm_envelope`). The number of simulated refits is taken from the option `visStatistics.qq_nsim` and defaults to 5000. As `visstat()` has no corresponding argument, this option is the only way to change it here; lower it to trade precision for speed, for instance `options(visStatistics.qq_nsim = 1000L)`. Use `vis_lm_assumptions` or `qq_lm_envelope` directly if you prefer to set the number of refits per call.

Value

An object of class "visstat" containing the results of the automatically selected statistical test. The specific contents depend on which test was performed. Additionally, the returned object includes two attributes:

- `plot_paths`: Character vector of file paths where plots were saved (if `graphicsoutput` was specified)
- `captured_plots`: List of captured plot objects for programmatic access

In case of insufficient data, returns a list with an error element and basic input summary information.

Note

For best visualization, ensure that the RStudio Plots pane is adequately sized. If you get "figure margins too large" errors, try expanding the Plots pane in RStudio, or using `dev.new(width=10, height=6)` for a larger plot window.

See Also

`visstat_core` defining the decision logic, the package's vignette `vignette("visStatistics")` explaining the decision logic accompanied by illustrative examples, and the accompanying webpage <https://shhschilling.github.io/visStatistics/>.

Examples

```
old_qq_nsim <- getOption("visStatistics.qq_nsim")
options(visStatistics.qq_nsim = 1000L)

# Formula interface
```

```

mtcars$am <- as.factor(mtcars$am)
visstat(mpg ~ am, data = mtcars)

# Standardised usage
visstat(mtcars$am, mtcars$mpg)

## Student's t-test (equal variances, two groups)
# When residuals are normally distributed and Levene's test indicates
# homoscedasticity, the classic Student's t-test with pooled variance is used
df <- droplevels(subset(PlantGrowth, group %in% c("ctrl", "trt1")))
visstat(df$group, df$weight)

## Welch's t-test (unequal variances, two groups)
# When residuals are normally distributed but Levene's test indicates
# heteroscedasticity, Welch's t-test is used
visstat(mtcars$am, mtcars$mpg)

## Wilcoxon rank sum test (non-normal, two groups)
# When residuals are not normally distributed
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)

## Fisher's ANOVA (equal variances, >2 groups)
# When residuals are normally distributed and Levene's test indicates
# homoscedasticity, classic Fisher's ANOVA with TukeyHSD post-hoc is used.
# Different green letters indicate significant differences between groups.
visstat(PlantGrowth$group, PlantGrowth$weight)

## Welch's one-way ANOVA (unequal variances, >2 groups)
set.seed(123)
values <- c(rnorm(20, 10, 1), rnorm(20, 15, 5), rnorm(20, 12, 2))
groups <- factor(rep(c("A", "B", "C"), each = 20))
visstat(groups, values)
## Kruskal-Wallis (non-normal, >2 groups)
# When residuals are not normally distributed, kruskal.test() is followed by
# pairwise.wilcox.test.
visstat(iris$Species, iris$Petal.Width)

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

## Pearson's Chi-squared test (both factors, large expected counts)
HairEyeColorDataFrame <- counts_to_cases(as.data.frame(HairEyeColor))
visstat(HairEyeColorDataFrame$Eye, HairEyeColorDataFrame$Hair, cex = 0.7)

```

```
## Fisher's exact test (both factors, small expected counts)
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())

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

options(visStatistics.qq_nsim = old_qq_nsim)
```

visstat_core

Automated Visualization of Statistical Hypothesis Testing

Description

`visstat_core()` implements the decision tree used by [visstat](#). It receives a `data.frame` and two column names, determines the corresponding analysis route, creates the diagnostic and result plots, and returns the selected test results as a `visstat` object.

Usage

```
visstat_core(
  dataframe,
  varsample,
  varfactor,
  conf.level = 0.95,
  correlation = FALSE,
  numbers = TRUE,
  minpercent = 0.05,
  group_test = NULL,
  graphicsoutput = NULL,
  plotName = NULL,
  plotDirectory = getwd(),
  plot_args = list()
)
```

Arguments

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

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
correlation	Logical. If FALSE (default), performs simple linear regression analysis with confidence and prediction bands. If TRUE, performs Spearman correlation analysis with trend line only (no regression interpretation).
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.
group_test	Optional character. For Route 1 only, NULL keeps the default assumption gates, "welch" forces Welch-type mean tests, but still displays the assumption-diagnostic plot and warns when residual normality is rejected, whereas "rank" forces Wilcoxon/Kruskal-Wallis rank tests without assessing the assumptions.
graphicsoutput	saves plot(s) of type "png", "jpeg", "pdf", "svg", "ps" or "tiff" in directory specified in plotDirectory. If graphicsoutput=NULL, no plots are saved. Any other value is not supported by Cairo(): it triggers a warning and no file is written.
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.
plot_args	Optional named list of base graphics parameters.

Details

The decision logic is organised into four routes. Route 1 handles a numeric response with a categorical predictor. By default, Route 1 uses residual-based test selection: Shapiro–Wilk on model residuals gates mean-based versus rank-based analysis, and Levene gates equal-variance versus Welch-type mean tests inside the mean branch. Above 5000 observations, where `shapiro.test()` is undefined, the Anderson–Darling test takes over as the residual-normality gate. Alternatively, `group_test = "welch"` forces Welch-type mean tests, and `group_test = "rank"` forces Wilcoxon/Kruskal–Wallis tests.

Route 2 handles ordered responses with categorical predictors by converting the ordered response to integer level codes and applying Wilcoxon or Kruskal–Wallis tests. Route 3 handles two numeric variables by fitting `lm()` by default, or Spearman rank correlation when `correlation = TRUE`. Route 4 handles two unordered factors with Pearson's χ^2 test or Fisher's exact test, depending on expected counts. If both variables are ordered and `correlation = TRUE`, Kendall's τ_b is used.

The significance level α is defined as $1 - \text{conf.level}$. Assumption tests are interpreted relative to this threshold.

Under the default `group_test = NULL`, Route 1 issues a warning when the group sizes are unbalanced, the largest group standard deviations occur in the smallest groups, and the route selected is either the equal-variance test or a rank-based test. In that configuration those two routes can exceed the nominal significance level, whereas a selected Welch test does not and is therefore not flagged; see the Route 1 simulations in `vignette("visStatistics")`.

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()` and `levene.test()` and `bp.test()`

For the general linear model the Shapiro-Wilk, Anderson-Darling, Levene and Bartlett tests are applied to the internally studentised residuals $r_i = e_i / (SE_{res} \sqrt{1 - h_i})$, which remove the leverage-dependent variance of the raw residuals ($Var(e_i) = \sigma^2 (1 - h_i)$).

Implemented post hoc tests:

- `TukeyHSD()` for `aov()`
- `games.howell()` for `oneway.test()`
- `pairwise.wilcox.test()` for `kruskal.test()`

The Q-Q envelopes in the assumption diagnostics are simulated (see [qq_lm_envelope](#)). The number of simulated refits is taken from the option `visStatistics.qq_nsim` and defaults to 5000. As `visstat_core()` has no corresponding argument, this option is the only way to change it here; lower it to trade precision for speed, for instance `options(visStatistics.qq_nsim = 1000L)`.

Value

An object of class "visstat" containing the results of the automatically selected statistical test. The specific contents depend on which test was performed. Additionally, the returned object includes two attributes:

- `plot_paths`: Character vector of file paths where plots were saved (if `graphicsoutput` was specified)
- `captured_plots`: List of captured plot objects for programmatic access

See Also

The package's vignette `vignette("visStatistics")` for a description of the decision logic, illustrated with numerous examples. The package is accompanied by its webpage <https://shhschilling.github.io/visStatistics/>. The main function `visstat` provides a detailed description of the return value.

Examples

```
old_qq_nsim <- getOption("visStatistics.qq_nsim")
options(visStatistics.qq_nsim = 1000L)

# 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")

## Simple 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")

options(visStatistics.qq_nsim = old_qq_nsim)

```

Description

Checks the residual diagnostics in the general linear model Student's t-test (`t.test, var=EQUAL`) Fisher oneway ANOVA (`aov`) or simple linear regression. Performs the Shapiro-Wilk and Anderson-Darling tests for normality, and for grouped data also Levene's and Bartlett's tests for homogeneity of variances. For simple linear regression, heteroscedasticity is assessed with the Breusch-Pagan test [[@Koenker:1981](#)], which regresses squared raw residuals on fitted values. The normality tests, the grouped variance tests, and the histogram and Q-Q panels are computed from the internally studentised residuals $r_i = e_i / (SE_{res} \sqrt{1 - h_i})$, which remove the leverage-dependent variance of the raw residuals ($Var(e_i) = \sigma^2 (1 - h_i)$). The residuals-vs-fitted panel (regression mode) uses the z-residuals $z_i = e_i / SE_{res}$, which retain the leverage-dependent spread.

Usage

```
vis_lm_assumptions(
  samples,
  fact,
  cex = 1,
  correlation = FALSE,
  conf.level = 0.95,
  qq_nsim = getOption("visStatistics.qq_nsim", 5000L),
  plot_args = list()
)
```

Arguments

<code>samples</code>	Numeric vector; the dependent variable.
<code>fact</code>	Factor; the independent variable.
<code>cex</code>	Numeric; scaling factor for plot text and symbols (default: 1).
<code>correlation</code>	Logical. If FALSE and fact is numeric, regression diagnostics are shown. If TRUE, no regression diagnostics are shown. Default is FALSE.
<code>conf.level</code>	Numeric confidence level for the simulated Q-Q envelopes.
<code>qq_nsim</code>	Integer number of simulated refits for the Q-Q envelopes.
<code>plot_args</code>	Optional named list of base graphics parameters.

Value

A list with elements:

summary_anova Summary of the ANOVA model.

shapiro_test Result from `shapiro.test()`.

ad_test Result from `nortest::ad.test()` or a character message if $n < 7$.

levene_test Result from `levene.test()` (grouped diagnostics only).

bartlett_test Result from `bartlett.test()` (grouped diagnostics only).

bp_test Result from `bp.test()` (regression diagnostics only).

Examples

```
ToothGrowth$dose <- as.factor(ToothGrowth$dose)
vis_lm_assumptions(ToothGrowth$len, ToothGrowth$dose, qq_nsim = 100L)
```

Index

bp.test, [3](#)

counts_to_cases, [4](#)

effect_size, [5](#)

games.howell, [8](#)

levene.test, [9](#)

openGraphCairo, [10](#)

plot.visstat, [12](#), [13](#)

print.visstat, [12](#), [13](#), [16](#)

qq_lm_envelope, [14](#), [19](#), [23](#)

saveGraphVisstat, [15](#)

summary.visstat, [12](#), [13](#), [16](#)

vis_lm_assumptions, [19](#), [24](#)

visstat, [12](#), [13](#), [16](#), [17](#), [21](#), [23](#)

visstat_core, [17](#), [19](#), [21](#)