# Package 'heteroTests'

November 2, 2026

**Title** Heteroscedasticity Diagnostics for Linear Models

**Version** 0.17.0

**Description** Provides a unified set of heteroscedasticity diagnostics for linear-model workflows. It implements classical auxiliary-regression tests, including those of White (1980) <doi:10.2307/1912934>, Breusch and Pagan (1979) <doi:10.2307/1911963>, Koenker (1981) <doi:10.1016/0304-4076(81)90062-2>, Goldfeld and Quandt (1965) <doi:10.1080/01621459.1965.10480811> and Harvey (1976) <doi:10.2307/1913974>; the score test of Cook and Weisberg (1983) <doi:10.1093/biomet/70.1.1>; the ARCH test of Engle (1982) <doi:10.2307/1912773>; and group-wise tests of equal variance, including those of Bartlett (1937) <doi:10.1098/rspa.1937.0109>, Brown and Forsythe (1974) <doi:10.1080/01621459.1974.10482955> and Hartley (1950) <doi:10.2307/2332383>. Resampling and scalable variants, simulation utilities, diagnostic visualisation and remediation helpers share a consistent interface designed for reproducible statistical workflows and integration with common modelling tools.

**License** Apache License (>= 2.0)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**Depends** R (>= 4.1)

**Imports** MASS, stats, ggplot2, curl, generics, scales, R6, parallel, SuppDists

**Suggests** quickcheck, testthat (>= 3.0.0), styler, lintr, digest, covr, knitr, rmarkdown, rjtools, gridExtra, shiny, DT, plotly, htmlwidgets, readxl, Matrix, bench, lmtest, plm, withr, car, vartest, mgcv, quantreg, sandwich, spdep, broom, workflows, parsnip, recipes, hardhat, survey, data.table, dtplyr, dplyr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**URL** https://github.com/DiogoRibeiro7/heteroTests, https://diogoribeiro7.github.io/heteroTests/

**BugReports** https://github.com/DiogoRibeiro7/heteroTests/issues

**NeedsCompilation** no

**Author** Diogo Ribeiro [aut, cre] (ORCID:
<https://orcid.org/0009-0001-2022-7072>)

**Maintainer** Diogo Ribeiro <dfr@esmad.ipp.pt>

**Repository** CRAN

**Date/Publication** 2026-11-01 23:10:02 UTC

# Contents

.diagnostic_registry . . . . . 4
additional_tests . . . . . 5
analyzeMLResiduals . . . . . 5
autoCompareRemediations . . . . . 6
autoplot.hetero_test_suite . . . . . 6
autoTransform . . . . . 7
bootstrapTestStatistic . . . . . 8
bootstrap_methods . . . . . 13
cachedTest . . . . . 13
checkData . . . . . 15
checkModel . . . . . 15
checkModelEnhanced . . . . . 16
clearAnalysisCache . . . . . 17
clearTestCache . . . . . 18
compareModelDiagnostics . . . . . 19
compareTestResults . . . . . 20
diagnostic_data . . . . . 20
downloadTeachingData . . . . . 21
fitRobust . . . . . 22
fitWLS . . . . . 23
generateDiagnosticReport . . . . . 26
generateHeteroRecommendations . . . . . 27
HeteroDiagnostic . . . . . 29
hetero_data . . . . . 29
hetero_test_suite . . . . . 30
ht_set_log_level . . . . . 30
launchDiagnosticDashboard . . . . . 31
listDiagnostics . . . . . 32
modern_diagnostics . . . . . 33
performArchLMTest . . . . . 33
performBartlettTest . . . . . 34
performBoxMTest . . . . . 36
performBPRandomEffectsTest . . . . . 38
performBPTest . . . . . 39
performBPTestRobust . . . . . 42
performBPTestStreaming . . . . . 44

performBrownForsytheTest . . . 46
performCookWeisbergTest . . . 47
performDavidianCarrollTest . . . 49
performFlignerKilleenTest . . . 50
performGlejserTest . . . 53
performGQTest . . . 56
performHarrisonMcCabeTest . . . 58
performHartleyFmaxTest . . . 62
performHarveyTest . . . 63
performHighDimensionalTest . . . 65
performInfluenceDiagnostics . . . 67
performKoenkerTest . . . 67
performKoenkerTestStreaming . . . 70
performLeveneTest . . . 72
performMcLeodLiTest . . . 73
performNCVTest . . . 74
performOBrienTest . . . 76
performParkTest . . . 78
performPesaranTest . . . 79
performQuantileRegressionTest . . . 81
performRankPermutationTest . . . 83
performRESETTest . . . 85
performScatterDiagnostic . . . 86
performSpatialHeteroTest . . . 88
performSpearmanTest . . . 89
performSpreadLevelTest . . . 91
performStudentizedBPTest . . . 92
performSzroeterTest . . . 94
performVarianceFormTest . . . 97
performVIFDiagnostic . . . 101
performWhiteTest . . . 102
performWhiteTestBootstrap . . . 106
performWhiteTestRobust . . . 110
performWhiteTestStreaming . . . 112
performWildBootstrapTest . . . 115
plot.HeteroDiagnostic . . . 119
plot.power_analysis . . . 119
plotBeforeAfter . . . 120
plotBubbleVariance . . . 121
plotDiagnosticSuite . . . 121
plotDiagnosticSuiteEnhanced . . . 122
plotResidualDensity . . . 122
plotResidualQQ . . . 123
plotResidualsFitted . . . 123
plotResidualsFittedEnhanced . . . 124
plotSpreadLevel . . . 125
print.htest_enhanced . . . 125
rcalculateEffectSize . . . 126

registerDiagnostic . . . 127
registerPlot . . . 127
restimate_test_power . . . 128
robust_implementations . . . 129
runAdvancedDiagnostics . . . 129
runDiagnosticPlots . . . 130
runDiagnostics . . . 131
runHeteroTests . . . 132
runHeteroTestsParallel . . . 136
runMultivariateTests . . . 137
runPanelTests . . . 137
runSurveyHeteroTests . . . 138
runTimeSeriesTests . . . 139
run_benchmark_suite . . . 140
rvalidate_against_reference . . . 142
scale_colour_hetero_diagnostic . . . 143
simulate_hetero . . . 144
simulation_framework . . . 146
std_error . . . 147
std_warning . . . 148
suggestRemediation . . . 148
summary.HeteroDiagnostic . . . 149
summary.hetero_test_suite . . . 150
test.HeteroDiagnostic . . . 154
TestFactory . . . 154
test_factory . . . 156
test_results . . . 156
theme_hetero . . . 163
tidy.hetero_test . . . 164
validateTestInputs . . . 165

**Index** **167**

---

`.diagnostic_registry` *Diagnostics registry*

---

## Description

Maintains a registry of diagnostic functions that can be called by `runHeteroTests()` and `runDiagnostics()`. Users can register custom diagnostics via `registerDiagnostic()`.

## Usage

```
.diagnostic_registry
```

## Format

An object of class `environment` of length 16.

## additional_tests *Additional heteroscedasticity tests*

### Description

These functions extend the package with more specialized tests for heteroscedasticity such as a studentized Breusch-Pagan test, a bootstrap version of White's test and the Szroeter ordered test.

## analyzeMLResiduals *Machine-learning residual analysis*

### Description

Fits a GAM using mgcv and compares its residuals to the input linear model.

### Usage

```
analyzeMLResiduals(model, data = NULL)
```

### Arguments

| | |
|---|---|
| model | A fitted lm object or a formula. |
| data | Data frame used if model is a formula. |

### Value

A list with the GAM model, residuals and RMSE reduction.

### See Also

compareModelDiagnostics

### Examples

```
data(mtcars)
analyzeMLResiduals(mpg ~ wt + qsec, mtcars)
```

`autoCompareRemediations` *Automatically compare remedial models*

## Description

Fits several remedial models for heteroscedasticity and compares their performance using AIC and residual RMSE. Currently evaluates weighted least squares and robust regression against the original model.

## Usage

```
autoCompareRemediations(model, data = NULL)
```

## Arguments

| | |
|---|---|
| `model` | Fitted `lm` model or formula. Since 0.17.0 a `MASS::rlm()` fit, an M-estimate, is refused, as by every test: its residuals are not least-squares residuals. |
| `data` | Optional data frame if `model` is a formula. |

## Value

A list with components `metrics`, `models`, and `best` indicating the recommended method.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, mtcars)
autoCompareRemediations(m)
```

`autoplot.hetero_test_suite` *Autoplot heteroscedasticity diagnostics*

## Description

Draws one horizontal bar per test. The length of a bar is the evidence against the null hypothesis, $-\log_{10} p$, on an axis labelled with the p-values themselves, so a smaller p-value gives a longer bar. A dashed line marks `alpha`; bars that pass it are drawn in the accent colour and the others in grey. The p-value of each test is printed beside its bar, so no reading depends on colour.

## Usage

```
## S3 method for class 'hetero_test_suite'
autoplot(object, ..., alpha = 0.05)

## S3 method for class 'hetero_grouped_suite'
autoplot(object, ..., alpha = 0.05)
```

### Arguments

| | |
|---|---|
| object | A hetero_test_suite or hetero_grouped_suite. |
| ... | Unused. |
| alpha | Significance level marked by the dashed line. Defaults to 0.05. |

### Details

Bars stop at $p = 10^{-4}$: extra length beyond that would only compress the tests that are near the threshold. Labels give the p-value to three decimals, and p < 0.001 below that. A test that failed has no bar and is labelled as such.

For a grouped suite there is one panel per group.

### Value

A ggplot object.

### Earlier versions

Up to 0.12.0 the bars were the p-values themselves on a linear axis from 0 to 1, so a test that rejected had no visible bar, and the highlighting of significant tests was never applied to a suite of more than one test.

### Examples

```
fit <- lm(mpg ~ wt + qsec, data = mtcars)
suite <- runHeteroTests(
  fit, mtcars,
  tests = c("white", "breusch_pagan", "koenker"),
  progress = FALSE
)
ggplot2::autoplot(suite)
```

---

## autoTransform *Automatic transformation helper*

---

### Description

Chooses between no transformation, log, square root and Box–Cox based on the Breusch-Pagan test statistic.

### Usage

```
autoTransform(model)
```

### Arguments

| | |
|---|---|
| model | A fitted model of class lm. |

## Value

A list with elements `model` (the transformed fit) and `method` indicating the chosen transformation.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
autoTransform(m)
```

---

`bootstrapTestStatistic` *Bootstrap a test statistic*

---

## Description

Provides a consistent wrapper around bootstrap resampling for test statistics produced by heteroscedasticity diagnostics. The routine refits the supplied model on each bootstrap replicate and evaluates `test_function` on it, returning the bootstrap distribution, a percentile interval, and—for null-imposed resampling—a p-value.

## Usage

```
bootstrapTestStatistic(
  test_function,
  model,
  data,
  B = 999,
  parallel = FALSE,
  ci_level = 0.95,
  resample = c("null", "pairs"),
  n_cores = NULL,
  progress = interactive(),
  ...,
  reestimate_weights = NULL
)

rbootstrap_test_statistic(
  test_function,
  model,
  data,
  B = 999,
  parallel = FALSE,
  ci_level = 0.95,
  resample = c("null", "pairs"),
  n_cores = NULL,
  progress = interactive(),
  ...,
```

```
  reestimate_weights = NULL
)
```

## Arguments

| | |
|---|---|
| test_function | A function that accepts (model, data, ...) and returns an object with a numeric statistic element (typically an htest result). |
| model | A fitted model of class lm, with or without weights, or, since 0.17.0, a glm with the Gaussian family and the identity link, the least-squares fit of lm() with its prior weights, whose replicates are those of that lm. Another glm is rejected: each replicate is refitted with least squares, which would discard its family and link. A Gaussian mgcv::gam() fit, a glm by class, is accepted since 0.17.0 when it is the least-squares fit of its design, as one whose smooth terms are all fixed (s(x, fx = TRUE)) or one without smooth terms, and its replicates are those of the lm() of that design; one with penalised smooth terms is refused, as every gam was before. |
| data | Data frame used when fitting model, holding the rows it was fitted to, or its model frame. Other rows are left out. See Details. |
| B | Integer, number of bootstrap replications. Defaults to 999. The p-value is (1 + #) / (B_eff + 1), so a test at level alpha is exact only when alpha * (B + 1) is an integer: with 999 it is at the 5% and 1% levels, and with 1000, the default before 0.15.0, at neither. |
| parallel | Logical, compute replicates in parallel when possible? See *Parallel replicates*. |
| ci_level | Confidence level for the percentile interval. |
| resample | Bootstrap strategy. "null" (the default) regenerates the response under homoscedasticity and yields a testable p_value; "pairs" samples rows of data with replacement and yields replicates and an interval but no p-value. See Details. |
| n_cores | Optional integer specifying the number of worker processes to use when parallel = TRUE. Defaults to parallel::detectCores() - 1. The result does not depend on it. |
| progress | Logical flag controlling whether a textual progress bar is displayed during bootstrap replication. |
| ... | Additional arguments forwarded to test_function. |
| reestimate_weights | Whether each replicate estimates the weights of a fitWLS() fit again from the variance function fitWLS() recorded on it. NULL, the default, does so for a fitWLS() fit and holds the weights fixed for any other model. FALSE holds them fixed, as every call did before 0.15.0. TRUE is an error for a model that fitWLS() did not return, or on which it fell back to equal weights. Each replicate regenerates the response and holds the design fixed, so a variance regressor that uses a variable of the response which the right-hand side of the model does not use, such as y or x for I(y / x) ~ z, or that is the response term itself, as ~ . gives, cannot be estimated again: NULL then holds the weights fixed, with a warning, and TRUE is an error. Giving the response as a column of the data, as in r ~ z, says which variables stay. It comes after ..., so it has to be named in full. See Details. |

## Details

The `resample` argument decides what the replicates mean.

With `resample = "null"` (the default) the response is regenerated as $\hat{y}_i + e_i^*$, where the $e_i^*$ are drawn with replacement from the model's residuals after dividing by $\sqrt{1 - h_i}$ and centring. Those data satisfy homoscedasticity by construction, so the replicates estimate the null distribution and `p_value` is a test. The leverage correction matters: OLS residuals have variance $\sigma^2(1 - h_i)$, and resampling them unscaled under-disperses the regenerated errors, which inflates the rejection rate.

With `resample = "pairs"` rows are sampled with replacement. The replicates then follow the statistic's distribution under whatever variance structure the data actually have, which is useful for describing its variability but is not a null distribution: the replicates are centred on the observed statistic, so comparing the two returns roughly 0.5 whatever the data. Measured rejection under $\sigma_i = x_i^2$ was 0%. `p_value` is therefore `NA` for pairs resampling.

Neither strategy accepts a `glm`: each replicate is refitted with least squares, so the family and link would be discarded and the replicates would describe a different model from the one supplied.

A weighted `lm` is resampled on the scale where its null hypothesis is constant variance, the regression of $\sqrt{w_i}\, y_i$ on $\sqrt{w_i}\, x_i$. Null-imposed resampling draws the $e_i^*$ from the Pearson residuals $\sqrt{w_i}\, e_i$, divided by $\sqrt{1 - h_i}$ with the leverages of that regression and centred, and divides each draw by $\sqrt{w_i}$ before adding it to $\hat{y}_i$. Pairs resampling draws each row with its weight. Either way each replicate is refitted with the weights, so `test_function` sees a weighted fit, as it did on the original data.

Weights given to the fit are known, and they are held fixed. On a fit returned by `fitWLS()` they were estimated from the same data, and since 0.15.0 each replicate estimates them again by default (`reestimate_weights`), so that `test_function` sees what `fitWLS()` would have returned for the replicate. The replicate is fitted as the model `fitWLS()` was given, without weights or with the weights that model had. The variance function `fitWLS()` recorded is fitted again to its residuals, with the same form and on the regressors `fitWLS()` evaluated, whose rows follow the rows drawn under pairs resampling, and the replicate is refitted with the weights it gives. A replicate whose variance function cannot be fitted, where `fitWLS()` would fall back to equal weights, fails as a replicate whose refit fails does, and the p-value and the interval are computed from the replicates that remain; a warning gives the number that failed. In `inst/validation/weighted-fits-size.R`, with the variance function correctly specified, normal errors, 150 observations, `B = 499` and `performKoenkerTest()` as `test_function`, the rejection rate at the 5% level was 0.083 with the weights held fixed, as every call did before 0.15.0 and `reestimate_weights = FALSE` still does, 0.045 with them estimated again, and 0.045 with the true weights. See the section on weighted fits in `performWildBootstrapTest()`.

Weighted or not, each replicate is refitted by least squares on the design stored with the fit, in the rows of the replicate, with the weights and the total offset of those rows, the `offset` argument and the `offset()` terms. Nothing is evaluated again from the formula or from `data`: a term computed from the rows it is given, such as `cut(x, quantile(x))`, the knots of `splines::ns()` or `I(x > median(x))`, keeps in every replicate the values the fit computed, and a variable the formula found outside `data`, such as `offset(o)` with `o` in its environment, follows the rows drawn. Under pairs resampling, a level of a factor or a value of a character regressor that no row drawn holds is left out of the replicate, as a refit of the formula leaves it out, and the contrasts of the fit are kept, but for a contrasts matrix of such a factor, which gives way to the default contrasts. `test_function` is given the replicate as an `lm`, whose model frame is that of the fit in the rows of the replicate with the response of the replicate, and the rows of `data`, regenerated or drawn. Up to 0.16.0 each replicate was refitted from the formula on the rows of `data`, and a fit whose model frame could not be rebuilt

from them so was refused ("data does not reproduce the fit"), as with cut(x, quantile(x)), a term computed from all the rows on a fit that dropped some, or a model that transforms a variable given its model frame as data; so was pairs resampling of a model with a variable outside data. The random numbers are drawn as before, in the same order, so a call made after set.seed() returns what it returned wherever the stored design is the design the formula rebuilt.

The rows of data are matched to those of the fit by row name, so that each row is regenerated, or drawn, with its own fitted value, weight and offset. Rows the model was not fitted to, such as those it dropped for missing values, are left out, for the observed statistic as for the replicates; the rows that stay keep the order of data. data that lacks a row the model was fitted to is refused. Data without row names of its own, such as a tibble or a data.table, that has as many rows as the fit is taken in the order of the fit. The rows are then matched to the observations of the fit as the tests match them (see test_results), and data whose rows cannot be shown to hold them are refused, as are values that differ from those the model was fitted to. A model whose least-squares refit of its design does not give its residuals, such as an lm fitted with a smaller tol that keeps a column lm() would alias, is refused too ("data does not reproduce the fit"): its replicates would be those of another model. Since 0.17.0 a MASS::rlm() fit, an M-estimate, is refused first, as by every test (see *Supported models*); it was refused here before, with that message.

Null-imposed resampling additionally requires the response to be a plain variable. For log(y) ~ x the fitted values are on the log scale while the data column holds y, so regenerating one from the other and refitting would take the logarithm twice. That case raises an error; resample = "pairs" resamples rows and is unaffected by it.

The warning White's, Koenker's and the Breusch-Pagan tests give since 0.16.0 on a factor with more levels than observations per level is about their asymptotic p-value (see *Fixed effects for small groups* in performKoenkerTest()). The observed statistic and the replicates are computed here without it, so that performWhiteTestRobust() and performBPTestRobust() with bootstrap = TRUE give it once, for the asymptotic p-value they report. The result's original, the test's own result, carries that asymptotic p-value: on such a design it is unreliable, and the bootstrap's, p_value, is the one to use.

## Value

A list with components:

original Result returned by test_function on the original data.

original_statistic Numeric value of the test statistic from the original sample.

replicates Numeric vector of bootstrap statistics.

ci Percentile interval of the bootstrap distribution of the statistic, at ci_level. This summarises where the resampled statistic falls; it is not a confidence interval for a parameter and carries no coverage guarantee.

p_value Empirical p-value from the null-imposed replicates, computed as $(1+\#\{T_b^* \geq T\})/(B_{\text{eff}}+1)$; NA when resample = "pairs".

B Requested number of replications.

effective_samples Number of non-missing bootstrap replications.

call Matched call for reproducibility.

### Parallel replicates

With `parallel = TRUE` replicate `i` draws its random numbers from stream `i` of a L'Ecuyer-CMRG generator (`parallel::nextRNGStream()`), seeded by one number drawn from the session's generator. A call made after `set.seed()` therefore returns the same result each time, whatever `n_cores` and whether the workers are forked (`parallel::mclapply()`) or run on a socket cluster (`parallel::makeCluster()`, where the platform cannot fork). It does not return the result of the same call with `parallel = FALSE`, which draws every replicate from the session's generator, as it did before 0.15.0: the two are bootstraps of the same data with different draws, and each is valid. Before 0.15.0 a seeded call with `parallel = TRUE` gave a different result each time.

### Deprecated name

`rbootstrap_test_statistic()` is deprecated since 0.14.0: use `bootstrapTestStatistic()`, the same function under the package's naming scheme. The old name still works, returns what it returned and warns.

### Supported models

It accepts an `lm`, unweighted or with positive weights, a `fitWLS()` fit, since 0.16.0 a rank-deficient `lm`, and since 0.17.0 a `glm` with the Gaussian family and the identity link, weighted or not, bootstrapped as the `lm` it is; a replicate whose refit aliases a coefficient the fit estimated, as a pairs resample of an interaction can, fails. Since 0.17.0 it accepts an `mgcv::gam()` fit that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`) or one without smooth terms, bootstrapped on that design, the basis of its smooths, as the `lm()` of it (decision 38 of the design of the model context); it refused every `gam` before. It refuses a `glm` of another family or link and a `survey::svyglm()` fit, as each replicate is refitted by least squares, an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, with the class of the refusals of the model context since 0.17.0 (decision 37), a fit with a zero weight, a `MASS::rlm()` fit, an M-estimate, with the message of every test since 0.17.0 (it was refused before as a fit the least-squares refit does not reproduce), a fit whose coefficients, residuals or weights were changed after it was fitted (since 0.17.0), a `fitWLS()` fit of an `mgcv::gam()` fit with smooth terms, whose terms do not rebuild the design of its coefficients (since 0.17.0), and `data` whose rows cannot be shown to hold the observations of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### Examples

```
data(mtcars)
model <- lm(mpg ~ wt + cyl, data = mtcars)
if (interactive()) {
  set.seed(123)
  boot <- bootstrapTestStatistic(performWhiteTest, model, mtcars, B = 100)
  boot$ci
}
```

## bootstrap_methods *Bootstrap utilities for heteroscedasticity diagnostics*

### Description

These helpers provide reusable bootstrap and power-estimation routines for the heteroscedasticity testing framework. They support the robust implementations introduced in `robust_implementations.R` but are written generically so additional tests can reuse them.

## cachedTest *Cache results of a heteroscedasticity test*

### Description

Runs a registered diagnostic, and returns the result of an earlier call on the same inputs instead when the diagnostic is deterministic.

### Usage

```
cachedTest(test_name, model, data, ..., use_cache = TRUE)
```

### Arguments

| | |
|---|---|
| `test_name` | Character scalar naming a diagnostic of the registry (`listDiagnostics()`) or of `test_factory`. |
| `model` | Fitted model object supplied to the diagnostic. |
| `data` | Data frame used to fit the model. |
| `...` | Additional arguments forwarded to the diagnostic. |
| `use_cache` | Logical scalar: may the result be read from the cache and stored in it? `FALSE` runs the diagnostic. Defaults to `TRUE`. |

### Details

**What is cached.** Only the deterministic tests the package registers: `"white"`, `"breusch_pagan"`, `"koenker"`, `"cook_weisberg"`, `"ncv"`, `"spread_level"`, `"student_bp"`, `"szroeter"`, `"quantile_regression"`, `"high_dimensional"` and `"variance_form"`.

**What is not.** A test that draws random numbers is never cached: `"wild_bootstrap"`, `"white_bootstrap"`, `"rank_permutation"` and `"spatial_hetero"` run on every call, so a seeded call returns what it returns without the cache, and leaves the random number generator where it leaves it without the cache. Nor is `"box_m"`, which tests data rather than a model, a diagnostic added with `registerDiagnostic()` or `test_factory$register()`, or one registered again under a name of the package: the package cannot know whether it draws random numbers or reads anything besides its inputs. A result whose computation moved the random number generator is not stored either.

Before 0.16.0 every name was cached, the resampling tests included: a call with another seed returned the replicates of the first, and the random numbers drawn after it were not those of a call without the cache.

**The key.** The test, its arguments, and the parts of the model and of the data the test reads: the class of the model, its family and link; from the model context of the fit, its model frame, response, design, residuals and fitted values, prior weights, total offset, rank and aliased coefficients, the names of its observations and its `na.action`; its coefficients, the weights it stores, its terms (with their `predvars`), call, contrasts and levels, residual degrees of freedom, whether it keeps its model frame, and its attributes, such as the variance function of a `fitWLS()` fit; and `data`, whole: its values, row names, class and attributes. The key is an xxhash64 digest of those values (from the **digest** package; without it nothing is cached). It holds values, not environments: a model refitted to the same data, in another environment or another session, has the same key and is answered from the cache, with the model of the call as `attr(result, "model")`, so that the result is `identical()` to the one the test computes. A model the tests refuse before reading it, such as a fit with `model = FALSE` whose data changed after the fit, has no key and is refused as without the cache; so has a fit with `model = FALSE` whose data are gone, whose model frame the tests rebuild from `data`, and it runs without the cache.

**What the key leaves out.** The key does not hold the environment of the formula: a test that evaluates the formula on `data` again, as the check of the rows of `data` does, reads the functions and variables of that environment as they are, so that after `f` is defined again a model `y ~ f(x)` is answered with the result computed with the old `f`, where without the cache the check may refuse `data`. Clear both caches, with `clearTestCache()` and `clearAnalysisCache()`, after changing one of them. Nor does it hold the options of the session: a result read from the cache gives none of the warnings and messages of the call that computed it but one (below), so a call made under `options(warn = 2)` after one that stored its result is answered with that result, where without the cache a warning of the test, turned into an error, stops it. The results of the tests otherwise do not depend on the options; the names of the estimates of the quantile regression test depended on `options(digits)` before 0.16.0.

Before 0.16.0 the key held only the test, the coefficients, the residuals, `data` and the arguments. Fits that share those shared one entry, and the second call returned the result of the first: `lm(y ~ x + offset(o))` and `lm(I(y - o) ~ x)`, which have the same coefficients and residuals and other fitted values; a fit and the same fit with its weights changed, which the tests refuse; or a fit with `model = FALSE` before and after its data changed, which they refuse too.

**The bound.** The cache holds at most `getOption("heteroTests.cache_size", 100)` results, and drops the one used least recently to store another; `options(heteroTests.cache_size = 0)` turns it off, together with the cache of `runHeteroTests()`. A stored result keeps no copy of the model. Before 0.16.0 the cache grew without bound, by about 25 KB per call on a new model, and kept about 0.3 KB per call after `clearTestCache()`. The checks the tests make first keep their results in a cache of their own, of at most `getOption("heteroTests.validation_cache_size", 1000)` results, which `0` turns off; it holds no check that drew random numbers.

A result read from the cache is returned without running the test, so the call gives none of the warnings and messages of the call that computed it, but one: since 0.16.0 the warning White's, Koenker's and the Breusch-Pagan tests give on a factor with more levels than observations per level, of class `heteroTests_fixed_effects_warning`, is stored with the result and given again on every call that reads it, as by `runHeteroTests()` (see *Fixed effects for small groups* in `performKoenkerTest()`). The call that computes the result gives it once the result is stored, so that under `options(warn = 2)` the warning stops the call after the result is stored.

### Value

The diagnostic result, either retrieved from cache or freshly computed.

### See Also

`clearTestCache()` empties the cache; `runHeteroTests()` uses it with `use_cache = TRUE`.

---

## checkData *Validate data argument (deprecated)*

---

### Description

Ensures that `data` is a data.frame. Used internally for input validation across the package.

### Usage

```
checkData(data)
```

### Arguments

| | |
|---|---|
| `data` | Object to check. |

### Details

Deprecated since 0.14.0: an internal helper of the package, which leaves the exports in 1.0.0. It still works and warns.

### Value

Invisible `data` if valid, otherwise an error is thrown.

---

## checkModel *Validate model argument*

---

### Description

Checks that `model` is of class `lm` or `glm`. With `enhanced = TRUE` it also runs the soft checks of the deprecated `checkModelEnhanced()`: it warns when the residual degrees of freedom fall below six, when a near-perfect fit is detected ($R^2 > 0.999$), or, when `data` is given, when a variance inflation factor exceeds 10 (Belsley et al., 1980).

### Usage

```
checkModel(model, data = NULL, enhanced = FALSE)
```

### Arguments

| | |
|---|---|
| model | An object to check. |
| data | Optional data.frame used when fitting model. With enhanced = TRUE it enables the multicollinearity check through performVIFDiagnostic(). |
| enhanced | Logical. FALSE, the default, checks the class only. |

### Details

It does not check whether a test accepts model: it accepts a binomial or a Poisson glm, a survey::svyglm() fit and a fit with a zero weight, which the tests refuse, each with its own message. Which test accepts which model is set out in test_results, under *Supported models*.

The R-squared is that of the least-squares fit: for a glm with the Gaussian family and the identity link, that of the equivalent lm, which summary() of a glm does not report; for any other glm it is not checked. Before 0.16.0, enhanced = TRUE stopped on any glm with "missing value where TRUE/FALSE needed".

### Value

Invisible model if valid, otherwise an error is thrown.

### References

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Wiley.

---

## checkModelEnhanced *Enhanced model diagnostics (deprecated)*

---

### Description

Deprecated since 0.14.0: use checkModel(model, data, enhanced = TRUE), which runs the same checks. This name still works and warns.

### Usage

```
checkModelEnhanced(model, data = NULL)
```

### Arguments

| | |
|---|---|
| model | Fitted model supplied to subsequent diagnostics. |
| data | Optional data.frame used when fitting model. When provided it enables multicollinearity checks via performVIFDiagnostic(). |

### Details

Extends checkModel() with additional soft checks that flag potential numerical issues before computationally expensive diagnostics are executed. The helper mirrors the warnings used throughout the package and retains the original return semantics for backward compatibility.

The function warns when residual degrees of freedom fall below six, when a near-perfect fit is detected ($R^2 > 0.999$), or when variance inflation factors (VIFs) exceed 10. The latter threshold follows the regression diagnostics literature (Belsley et al., 1980).

### Value

Invisibly returns `model` after issuing any relevant warnings.

### References

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Wiley.

### See Also

checkModel(), performVIFDiagnostic(), validateTestInputs()

### Examples

```
data(mtcars)
fit <- stats::lm(mpg ~ wt + hp, data = mtcars)
checkModel(fit, enhanced = TRUE)
```

---

## clearAnalysisCache *Clear cached diagnostic run results*

---

### Description

Removes the results of whole calls that runHeteroTests stored with use_cache = TRUE, so that later calls run their tests again.

### Usage

```
clearAnalysisCache()
```

### Details

A call is stored only when every test it runs is deterministic, under a key that holds the tests, whether they are streamed and in chunks of what size, and every part of the model and the data they read (see cachedTest). The cache holds at most `getOption("heteroTests.cache_size", 100)` calls and drops the one used least recently; `0` turns it off. Each test of the call is also kept in the cache of cachedTest, which clearTestCache empties.

## Value

NULL, invisibly.

## See Also

runHeteroTests, clearTestCache

---

clearTestCache *Clear cached test results*

---

## Description

Removes every result from the cache of cachedTest(), which runHeteroTests() also uses for each test with use_cache = TRUE, so that later calls compute them again.

## Usage

```
clearTestCache()
```

## Details

The cache is bounded and holds no test that draws random numbers (see cachedTest()), so clearing it is needed neither to bound the memory of a long session nor to draw new replicates with another seed. It is needed after a change the key does not see, such as a function or a variable of the environment of a model's formula; call clearAnalysisCache() then too, since the cache of whole calls of runHeteroTests() keeps the results of such a model as well.

## Value

NULL, invisibly.

## See Also

cachedTest() for invoking diagnostics with memoisation.

compareModelDiagnostics *Compare diagnostics across models*

## Description

Runs `runDiagnostics` for multiple models and collects test statistics. Diagnostics that cannot be computed (for example, due to validation failures) emit warnings and fill the corresponding entries with `NA_real_`. The original error messages are preserved on the returned data frame via the `diagnostic_errors` attribute.

## Usage

```
compareModelDiagnostics(models, data = NULL, tests = c("white", "koenker"))
```

## Arguments

| | |
|---|---|
| `models` | List of fitted models or formulas. |
| `data` | Optional data frame when formulas are supplied. |
| `tests` | Diagnostics to run. Defaults to the White and Koenker tests, the default of `runHeteroTests`; it was the White and the classical Breusch-Pagan tests up to 0.14.0. |

## Value

A data frame of statistics, one row per model. Failed diagnostics are represented by `NA_real_` entries and the underlying error messages can be retrieved from the `diagnostic_errors` attribute.

## See Also

`analyzeMLResiduals`

## Examples

```
data(mtcars)
m1 <- lm(mpg ~ wt + qsec, mtcars)
m2 <- lm(mpg ~ wt + hp, mtcars)
compareModelDiagnostics(list(m1, m2))
```

## compareTestResults *Summarise heteroscedasticity test results*

### Description

Runs selected diagnostics for a single model and returns a tidy data frame of test statistics and p-values.

### Usage

```
compareTestResults(model, data = NULL, tests = c("white", "koenker"))
```

### Arguments

| | |
|---|---|
| model | A fitted lm model or a formula. |
| data | Data frame used if model is a formula. |
| tests | Character vector of heteroscedasticity tests. Defaults to the White and Koenker tests, the default of runHeteroTests(); it was the White and the classical Breusch-Pagan tests up to 0.14.0. |

### Value

Data frame with columns test, statistic, and p.value.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, mtcars)
compareTestResults(m)
```

## diagnostic_data *Simulated dataset with collinearity and mild nonlinearity*

### Description

This dataset contains two predictors with correlation and a response including a quadratic term. It is useful for testing diagnostics beyond heteroscedasticity.

### Usage

```
data(diagnostic_data)
```

### Format

A data frame with 150 rows and 3 variables:

**x1** first predictor

**x2** second predictor, correlated with x1

**y** response

### Source

Simulated via set.seed(123).

### Examples

```
data(diagnostic_data)
head(diagnostic_data)
```

---

## downloadTeachingData *Download example teaching dataset*

---

### Description

Provides convenient access to example datasets hosted online. Currently supports the 'tips' dataset used in various tutorials.

### Usage

```
downloadTeachingData(
  name = "tips",
  destfile = tempfile(fileext = ".csv"),
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| name | Name of the dataset to download. Only "tips" is currently supported. |
| destfile | Optional path to save the downloaded dataset. Defaults to a temporary file. |
| quiet | Logical; if FALSE, informative messages are printed during the download process. |

### Value

Path to the downloaded dataset on success, or NULL invisibly if the download fails or no internet connection is available.

## Examples

```
# Online example (requires internet)
if (curl::has_internet()) {
  data_path <- downloadTeachingData(quiet = TRUE)
  if (!is.null(data_path)) {
    # Use downloaded data
  }
}


# Offline example using built-in data
data(mtcars)
model <- lm(mpg ~ wt + hp, data = mtcars)
result <- performWhiteTest(model, mtcars)
print(result)
```

---

`fitRobust` *Robust regression wrapper*

---

## Description

Provides a simple interface to `MASS::rlm` for robust estimation.

## Usage

```
fitRobust(model, data = NULL, ...)
```

## Arguments

| | |
|---|---|
| `model` | A fitted model of class `lm` or a formula. |
| `data` | Optional data frame if `model` is a formula. |
| `...` | Additional arguments passed to `MASS::rlm`. |

## Details

The result is an M-estimate, not a least-squares fit: its residuals are not least-squares residuals, and the null distributions of the tests of the package assume least squares. Since 0.17.0 every test refuses it, and so do `fitWLS()`, `runHeteroTests()` and `runDiagnostics()`, with an error of class `heteroTests_model_context_error`; until then 37 tests took it for an `lm` and tested its residuals as least-squares ones. To test the variance of the errors, test the `stats::lm()` fit of the same formula. The result is for estimation, and for comparisons such as those of `autoCompareRemediations()`, which reads its AIC and residuals only.

Since 0.17.0 the call the result records names what was given: the formula, and the data as the caller named them, or, for an `lm`, the `data`, `subset` and `na.action` of its call, so that `print()` and `summary()` show `MASS::rlm(formula = mpg ~ wt + qsec, data = mtcars)`, where they showed `rlm(formula = form, data = data)`, `update()` refits it, and the refusal of the tests names the `stats::lm()` call to test instead, `lm(mpg ~ wt + qsec, data = mtcars)`.

## Value

An object of class `rlm`.

## Examples

```
data(mtcars)
m <- fitRobust(mpg ~ wt + qsec, mtcars)
summary(m)
# The tests refuse it; test the lm() fit of the same formula instead.
performKoenkerTest(lm(mpg ~ wt + qsec, mtcars))
```

---

`fitWLS` *Weighted Least Squares wrapper*

---

## Description

Refit a model by feasible generalised least squares, weighting each observation by the inverse of an estimated error variance.

## Usage

```
fitWLS(
  model,
  data = NULL,
  var_formula = NULL,
  form = c("exponential", "power")
)
```

## Arguments

| | |
|---|---|
| `model` | A fitted model of class `lm`, or a `glm` with the Gaussian family and the identity link. Since 0.17.0 it refuses a `MASS::rlm()` fit, an M-estimate, whose residuals are not the least-squares residuals it fits its variance function to, and an `mgcv::gam()` fit with penalised smooth terms, which it refitted by weighted least squares on the basis of its smooths without the penalty; a `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is refitted on that design (decisions 4, 37 and 38 of the design of the model context). |
| `data` | Optional base::data.frame holding the variables named in `var_formula`. It is needed only when they are not in the model frame. Since 0.16.0 its rows are matched to the observations of the fit as the tests that read `data` match them, data with row names of their own in any order and data that number their rows (a tibble) in the order of the fit, and other data are refused, where numbered data in another order were read by position. A variable of `var_formula` that neither `data` nor the model frame holds is refused, where it was found in the workspace. Without `var_formula`, `data` is not read, but it is matched in the same way, and other data are refused, where they were ignored. |

`var_formula` One-sided formula naming the variance regressors, for example `~ x`. `NULL`, the default, uses the regressors of `model`.

`form` Character scalar choosing the variance function: `"exponential"` (the default) or `"power"`. See the section on variance functions.

## Details

The variance is *modelled*, not read off the residuals directly. A single squared residual is a one-degree-of-freedom estimate of $\sigma_i^2$ and far too noisy to invert: weighting by $1/e_i^2$ hands almost all of the weight to whichever observations the initial fit happened to reproduce most closely. This function instead fits a variance function to the squared residuals and takes $\hat{\sigma}_i^2$ from its fitted values, the standard feasible-GLS recipe; weights are $1/\hat{\sigma}_i^2$.

## Value

A new `lm` object fitted with weights. The estimated variances are attached as the `"variance_model"` attribute, one per row, as `weights()` of the result gives the weights: on a fit with `na.action = na.exclude`, `NA` at the rows the fit dropped (since 0.17.0; before, one per observation of the fit). The fitted variance function (its form, regressors and coefficients) is attached as the `"variance_function"` attribute, which is what `performVarianceFormTest()` reads, and what the bootstrap tests read to estimate the weights again in each replicate. Its `regressors`, the variance regressors on the scale of the form, and `index`, the fitted log-variance $\hat{\gamma}_0 + z_i^\top \hat{\gamma}$, have one row, or value, per observation of the fit, also on a fit with `na.action = na.exclude`, whose rows they are not padded to. The estimated variances are proportional to the exponential of `index`, unless the variance function could not be used and the fit fell back to equal weights, which the record says with `usable = FALSE` and its `reason`; `index` is then the fitted log-variance that was refused, or `NA`. When `model` was fitted with weights, the attribute also holds them, as `initial_weights`, one per observation of the fit too.

## Variance functions

`form` chooses the shape of the variance function and `var_formula` the variables $z_i$ it depends on. Both forms are linear in the logarithm of the variance and carry their own constant.

`"exponential"` $\log \sigma_i^2 = \gamma_0 + z_i^\top \gamma$, the multiplicative model of Harvey (1976), estimated by regressing $\log e_i^2$ on $z_i$. The log scale keeps the fitted variances positive without constraining the auxiliary regression. This is the default, and with `var_formula = NULL` it is what `fitWLS()` has computed since 0.9.0.

`"power"` $\log \sigma_i^2 = \gamma_0 + \sum_j \delta_j \log z_{ij}$, that is $\sigma_i^2 \propto \prod_j z_{ij}^{\delta_j}$, the model of Park (1966). It is the exponential form in the logarithms of the regressors, which must be strictly positive. $\delta = 2$ is the textbook case of a standard deviation proportional to $z$.

Other log-linear shapes are written through `var_formula`, for example `~ x + I(x^2)` or `~ sqrt(x)`. The additive model $\sigma_i^2 = \alpha_0 + z_i^\top \alpha$ is not offered: nothing keeps its fitted variances positive, and in the validation study it was not usable in a material share of samples drawn from an additive model itself (`inst/validation/README.md`).

Residuals that are numerically zero are floored before the logarithm is taken, with a warning.

Weights estimated this way are consistent under a correctly specified variance function, so standard errors from the returned fit are usable. They were not before 0.9.0, when the weights were the raw

inverse squared residuals: the weighted residual sum of squares then collapsed towards $n$ regardless of the data, and nominal 95% intervals covered the truth about 10% of the time.

Whether the variance function *is* correctly specified can be tested: pass the result to `performVarianceFormTest()`.

A test of constant variance on the result takes its weights as known, although they were estimated from the same data, so its reference distribution is only approximate: see the validation study in `inst/validation/README.md`. The bootstrap tests `performWildBootstrapTest()`, `performWhiteTestBootstrap()` and `bootstrapTestStatistic()` allow for the estimation. Since 0.15.0 they run `fitWLS()` again on each replicate by default: they fit it as `model` was fitted, fit the recorded variance function again to its residuals, on the recorded regressors, and refit it with the weights that gives. Their argument `reestimate_weights = FALSE` holds the weights fixed instead, as they did before 0.15.0.

If the variance function cannot be fitted, or yields no usable variation, the function falls back to equal weights, which reduces the result to the original OLS fit.

The weighted fit keeps the design and the offset of `model`, whether the offset was given as an argument, `lm(..., offset = o)`, or written in the formula, `offset(o)`. Its coefficients, fitted values and residuals are those `lm()` returns for the same call with the estimated weights added. On a model fitted with weights, the refit uses the estimated weights in their place, and `weights()` of the result returns them. Since 0.16.0 the response, the design and the offset are read from the fit, as the tests read them, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, is refused, where it was refitted to the response of the changed data; a fit with `model = FALSE` whose data are gone is refitted from `data` that give its response and fitted values.

A `glm` with the Gaussian family and the identity link is a least-squares fit, with the design and residuals of the `lm`, and since 0.16.0 the result for it is the `lm` returned for that `lm`, prior weights included, with the call of that `lm`: `stats::lm()`, without the arguments only `glm()` takes, so that `update()` fits an `lm` again. Before, it kept the class and the call of the `glm` without its family, which `summary()` could not read and every test refused. Any other `glm` is refused.

## References

Harvey, A. C. (1976). Estimating regression models with multiplicative heteroscedasticity. *Econometrica, 44*(3), 461–465. doi:10.2307/1913974

Park, R. E. (1966). Estimation with heteroscedastic error terms. *Econometrica, 34*(4), 888. doi:10.2307/1910108

## See Also

`performVarianceFormTest()` to test the variance function that was fitted.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
wls <- fitWLS(m)
summary(wls)

# Variance proportional to a power of one regressor
wls_power <- fitWLS(m, var_formula = ~ wt, form = "power")
attr(wls_power, "variance_function")$coefficients
```

`generateDiagnosticReport` *Automated diagnostic report generation*

## Description

Creates a standalone report summarizing heteroscedasticity tests and diagnostic plots for a fitted model.

## Usage

```
generateDiagnosticReport(
  model,
  data = NULL,
  output_format = "html",
  output_file = NULL,
  include_remediation = TRUE,
  include_theory = FALSE
)
```

## Arguments

| | |
|---|---|
| `model` | Fitted `lm` model or formula. |
| `data` | Optional data frame if `model` is a formula. |
| `output_format` | One of `"html"`, `"pdf"`, or `"word"`. |
| `output_file` | Path to write the report to. If `NULL`, a name is generated automatically. |
| `include_remediation` | Logical; include remediation suggestions if `TRUE`. |
| `include_theory` | Logical; include background theory section. |

## Value

Invisibly returns the path to the generated report.

## Examples

```
if (requireNamespace("rmarkdown", quietly = TRUE) &&
    rmarkdown::pandoc_available()) {
  model <- lm(mpg ~ wt + hp, data = mtcars)
  generateDiagnosticReport(
    model, mtcars,
    output_file = file.path(tempdir(), "diagnostic_report.html")
  )
}
```

generateHeteroRecommendations
*Intelligent recommendations for heteroscedasticity diagnostics*

## Description

These helpers assemble an end-to-end recommendation workflow that profiles the input dataset, selects appropriate heteroscedasticity diagnostics, interprets results with qualitative confidence labels, and suggests remediation strategies with decision-tree style guidance.

## Usage

```
analyseDatasetCharacteristics(data, response = NULL)

suggestDiagnosticsForProfile(profile)

interpretHeteroTestResults(test_results, alpha = 0.05)

recommendRemediationStrategies(test_results, model, data, profile = NULL)

warnDiagnosticAssumptions(profile, interpretations, model, data, test_results)

buildDiagnosticDecisionTree(profile, recommendations)

composeDiagnosticNarrative(profile, interpretations, remediation,
  assumption_warnings, decision_tree)

generateHeteroRecommendations(model, data = NULL, test_results = NULL,
  alpha = 0.05, include_report = TRUE)
```

## Arguments

| | |
|---|---|
| data | A data.frame containing the modelling variables. For recommendRemediationStrategies() and generateHeteroRecommendations(), its rows are matched to the observations of the fit as the tests match them, and those rows, in the order of the fit, are read by the tests and the advice, which read the residuals, grouped data keeping their groups (the profile of generateHeteroRecommendations(), its size and missing values, describes data as given): rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation; the rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused (see test_results). Before 0.17.0 the i-th row was read with the i-th observation: named data in another order than the fit gave the advice of other rows, and the full data of a fit that dropped rows were refused. |
| response | Optional response variable name used for labelling. |

| | |
|---|---|
| profile | Dataset profile produced by analyseDatasetCharacteristics(). |
| test_results | Named list of htest objects, typically returned by runHeteroTests(). |
| alpha | Significance level applied when interpreting p-values. |
| model | Fitted lm, or glm with the Gaussian family and the identity link, or model formula. The tests refuse a glm of any other family or link since 0.15.0. |
| interpretations | Output from interpretHeteroTestResults(). |
| recommendations | Output from suggestDiagnosticsForProfile(), used to build the decision tree. |
| remediation | Output from recommendRemediationStrategies(). |
| assumption_warnings | Character vector of generated warnings. |
| decision_tree | Data frame returned by buildDiagnosticDecisionTree(). |
| include_report | Logical; include a plain-language narrative in the output. |

## Details

The recommendation engine analyses dataset characteristics such as sample size, missingness, and predictor skewness to propose diagnostics tailored to the context. It automatically interprets test outcomes, attaches qualitative confidence labels, warns when assumptions are at risk, and suggests remediation strategies matched to detected variance patterns. The generated decision tree and narrative target non-specialist audiences who need actionable guidance.

## Value

analyseDatasetCharacteristics() returns a list describing dataset size, missingness, skewness, and risk factors. suggestDiagnosticsForProfile() returns a data frame of recommended diagnostics with rationales. interpretHeteroTestResults() yields a tidy data frame of interpretations and an overall summary. recommendRemediationStrategies() produces a structured remediation plan. warnDiagnosticAssumptions() returns warnings about assumption violations. buildDiagnosticDecisionTree() constructs a tidy decision tree. composeDiagnosticNarrative() outputs a plain-language character vector, and generateHeteroRecommendations() returns an object of class hetero_recommendation_report bundling all components.

## See Also

runHeteroTests(), suggestRemediation(), generateDiagnosticReport()

## Examples

```
model <- lm(mpg ~ wt + hp, data = mtcars)
recs <- generateHeteroRecommendations(model, mtcars)
print(recs)
```

## HeteroDiagnostic *Create a diagnostic object*

### Description

Create a diagnostic object that exposes `test()`, `plot()` and `summary()` methods. Formulas are accepted for convenience.

### Usage

```
HeteroDiagnostic(model, data = NULL)
```

### Arguments

| | |
|---|---|
| `model` | A fitted `lm`, or a `glm` with the Gaussian family and the identity link, or a formula. The tests refuse a `glm` of any other family or link since 0.15.0. |
| `data` | Data used to fit the model when `model` is a formula. |

### Value

An object of class `HeteroDiagnostic`.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
d <- HeteroDiagnostic(m, mtcars)
runDiagnostics(d$model, d$data)
```

## hetero_data *Simulated dataset with heteroscedastic errors*

### Description

This dataset is generated from the model $y_i = 1 + 2x_i + \varepsilon_i$, where $\varepsilon_i \sim N(0, (0.5 + 2x_i)^2)$. A fixed seed ensures repeatable values.

### Usage

```
data(hetero_data)
```

### Format

A data frame with 100 rows and 2 variables:

**x** predictor

**y** response

## Source

Simulated with `set.seed(42)`; `runif()` and `rnorm()`.

## Examples

```
data(hetero_data)
plot(hetero_data$x, hetero_data$y)
```

---

`hetero_test_suite` *S3 containers for heteroscedasticity diagnostics*

---

## Description

Collections of heteroscedasticity test results returned by `runHeteroTests`. A `hetero_test_suite` is a simple named list of `htest` objects with additional metadata. When diagnostics are evaluated on grouped data, the returned object inherits from `hetero_grouped_suite` and contains one `hetero_test_suite` per group along with the grouping keys.

## Details

These classes are primarily useful for method dispatch (e.g. the broom tidiers and `autoplot()` methods). Users typically interact with the objects via `print()` and `summary()` (see `summary.hetero_test_suite`), `generics::tidy()`, `generics::glance()`, `generics::augment()` and `ggplot2::autoplot()`.

## Attributes

A `hetero_test_suite` carries `attr(, "tests")`, the names asked for; `attr(, "model")`, the model tested; `attr(, "data_source")`, the class of the data, or `NA`; and, since 0.16.0, `attr(, "model_origin")` when `runHeteroTests` fitted the model itself: `"formula"` when it was given a formula, `"group"` for the refit of the model within a group of a grouped run, and `"survey"` for the fit to the data of a survey design (`runSurveyHeteroTests`). The call of such a model names variables of the runner rather than the caller's, so `summary()` shows its formula instead. A suite of a model you fitted has no `model_origin`. A `hetero_grouped_suite` carries `attr(, "group_keys")` and `attr(, "tests")`.

---

`ht_set_log_level` *Control logging for heteroTests diagnostics*

---

## Description

These helpers manage the package-level logging subsystem used to debug complex diagnostic workflows. Users can adjust verbosity, enable log capture, inspect recent log entries and clear the stored history when finished.

### Usage

```
ht_set_log_level(level = c("INFO", "WARN", "ERROR", "SILENT"))

ht_enable_log_capture(enabled = TRUE, max_entries = 1000L, sink = NULL)

ht_log_history()

ht_clear_log_history()
```

### Arguments

| | |
|---|---|
| level | Character string specifying the minimum level to emit. Accepted values are "INFO", "WARN", "ERROR" and "SILENT". |
| enabled | Logical flag, TRUE to enable capture and FALSE to disable. |
| max_entries | Maximum number of entries to retain in memory. Older entries are discarded first. Use Inf to keep all entries. |
| sink | Optional file path or connection to mirror log output. |

### Value

ht_set_log_level() returns the previous log level invisibly. ht_enable_log_capture() and ht_clear_log_history() return NULL. ht_log_history() returns a data frame with columns timestamp, level and message.

### Deprecated

ht_enable_log_capture(), ht_log_history() and ht_clear_log_history() are deprecated since 0.14.0: of the logging functions only ht_set_log_level() stays exported after 1.0.0. They still work and warn.

---

## launchDiagnosticDashboard

*Interactive diagnostic dashboard*

---

### Description

Launch a production-ready Shiny dashboard for running heteroscedasticity diagnostics, exploring interactive plots, and experimenting with simulation studies.

### Usage

```
launchDiagnosticDashboard(model, data)
```

### Arguments

| | |
|---|---|
| model | Fitted lm model. |
| data | Data frame used to fit the model. |

## Details

Requires optional packages **shiny**, **DT**, and **plotly**. Excel uploads additionally depend on **readxl** and exporting interactive graphics relies on **htmlwidgets**. The dashboard guides users through data preparation, model fitting, diagnostic testing, interactive Plotly visualisations, and a real-time simulation lab for exploring heteroscedastic data-generating processes. Download buttons are provided for both results tables and interactive plots.

## Value

A `shiny.appobj` that can be run with `shiny::runApp()`.

## Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE) &&
    requireNamespace("DT", quietly = TRUE) &&
    requireNamespace("plotly", quietly = TRUE)) {
  mod <- lm(mpg ~ wt + hp, data = mtcars)
  launchDiagnosticDashboard(mod, mtcars)
}
```

---

# `listDiagnostics` *List the registered diagnostics*

---

## Description

Returns the names `runHeteroTests()` accepts in `tests`: the diagnostics the package registers and any added with `registerDiagnostic()`.

## Usage

```
listDiagnostics()
```

## Details

`"box_m"` is registered for `runMultivariateTests()`, which calls it with data and a grouping factor rather than with a model.

## Value

A character vector of the registered names, sorted.

## See Also

`registerDiagnostic()`, `runHeteroTests()`

## Examples

```
listDiagnostics()
```

## modern_diagnostics *Modern heteroscedasticity diagnostics*

### Description

Collection of advanced heteroscedasticity tests including wild bootstrap, heteroscedasticity-consistent covariance LM statistics, quantile regression comparisons, rank-based permutation checks, high-dimensional projections and spatial correlation diagnostics.

### Usage

```
modern_diagnostics
```

### Format

An object of class `NULL` of length 0.

## performArchLMTest *Perform Engle's ARCH LM test*

### Description

Regresses squared residuals on their lags to detect ARCH effects.

### Usage

```
performArchLMTest(model, data = NULL, lags = 1)
```

### Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | optional data frame the model was fitted to. When `NULL`, the default, the model frame of `model` is used. The statistic is computed from the residuals of `model`, so `data` does not change the result; it is accepted so that every test takes `(model, data, ...)`. Before 0.15.0 the second argument was `lags`: a number passed there by position is still read as `lags`, with a warning that shows the new call; that form stops working in 1.0.0. |
| `lags` | number of lags to include. |

### Details

The statistic is $nR^2$ from an auxiliary regression of $e_t^2$ on its lagged values, where $R^2$ is the coefficient of determination. Under the null of no ARCH effects it follows a chi-square distribution with degrees of freedom equal to the number of lags.

## Value

An object of class `htest` containing the test statistic, p-value and degrees of freedom.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate `lm` ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without `data`. A fit with `model = FALSE` whose data are gone is tested on `data` that give its response and fitted values, and refused without them. The test reads nothing from `data`, and since 0.16.0 matches it to the observations of the fit as every test does: data with row names of their own in any order, data that number their rows (a tibble, a data.table) in the order of the fit; other numbered data, rows in front of those of the fit and repeated row names are refused, where it gave the value of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4), 987–1007. doi:10.2307/1912773

Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performArchLMTest(m, lags = 2)
```

---

# `performBartlettTest` *Perform Bartlett's test for equality of variances*

---

## Description

Uses `bartlett.test` on model residuals grouped by a factor.

## Usage

```
performBartlettTest(model, data = NULL, group)
```

## Arguments

| | |
|---|---|
| model | an object of class lm. |
| data | The data frame model was fitted to, or NULL, the default. The residuals and weights are read from the fit itself, and group from the rows of data that hold the observations of the fit; the observations whose group is missing are left out, with a warning. Without data it is read from the model frame of model, which holds it only when it is a variable of the model under its own name (x, not log(x)); a grouping variable the model does not use needs data. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| group | name of the grouping variable. |

## Details

The test statistic compares the pooled variance to the individual group variances. It is computed as $\chi^2 = (N-k)\ln S_p^2 - \sum(n_i-1)\ln s_i^2$, where $S_p^2$ is the pooled variance and $s_i^2$ the group variances. Under the null it approximates a chi-square distribution with $k-1$ degrees of freedom.

## Value

An object of class htest containing the chi-squared statistic, p-value and degrees of freedom.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, and data that lack rows or variables of the fit. It refuses a multivariate lm ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An mgcv::gam() fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context); since 0.16.0 its observations are matched to the rows of data by the row names of its model frame. Before, they were matched by position, since the residuals of a gam carry no names, and data with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## References

Bartlett, M. S. (1937). Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London*, 160(901), 268–282. doi:10.1098/rspa.1937.0109

Hartley, H. O. (1950). The maximum F-ratio as a short-cut test for heterogeneity of variance. *Biometrika*, 37(3/4), 308–312. doi:10.2307/2332383

## Examples

```
data(mtcars)
mtcars$cyl <- factor(mtcars$cyl)
m <- lm(mpg ~ wt, data = mtcars)
performBartlettTest(m, mtcars, "cyl")
```

---

`performBoxMTest` *Box's M Test for Equality of Covariance Matrices*

---

## Description

Test whether multiple groups have equal covariance matrices, by Box's (1949) chi-square approximation to his M statistic.

## Usage

```
performBoxMTest(data, group)
```

## Arguments

| | |
|---|---|
| `data` | A numeric data frame or matrix. |
| `group` | A factor indicating group membership. At least two groups must have observations, and each of them more observations than `data` has variables. |

## Details

`performBoxMTest(data, group)` is the one test that does not take `(model, data, ...)`: it tests the covariance matrices of the data themselves, not the residuals of a fitted model.

With $g$ groups of $n_i$ observations of $p$ variables, $\nu_i = n_i - 1$, $\nu = \sum_i \nu_i$, $S_i$ the covariance matrix of group $i$ and $S$ the pooled one, Box's statistic is

$$M = \nu \log |S| - \sum_i \nu_i \log |S_i|,$$

his correction factor is

$$c_1 = \left( \sum_i \frac{1}{\nu_i} - \frac{1}{\nu} \right) \frac{2p^2 + 3p - 1}{6(p+1)(g-1)},$$

and $M(1 - c_1)$ is referred to a chi-square distribution with $f_1 = (g-1)p(p+1)/2$ degrees of freedom: Box's (1949) chi-square approximation. The test assumes that each group is multivariate normal. It gives the values of `biotools::boxM()`, `heplots::boxM()` and `rstatix::box_m()`: 140.9430 on 20 degrees of freedom for the example on `iris`.

Before 0.16.0 the correction factor was $(g-1)c_1$: the same for two groups, whose values do not change, and too large for three or more, which made the statistic too small and the test conservative. The example on `iris` gave 135.2229.

The groups are the levels of `group` that have observations: a level with none, as `subset()` leaves on a factor, is dropped (since 0.16.0; before, it made the result `NA`). There must be at least two of

them, and each group needs more observations than there are variables, $n_i > p$: a group of $n_i \leq p$ observations has a singular covariance matrix (a group of one has none), whose log-determinant is $-\infty$, and the statistic is not defined, whatever the data. Since 0.16.0 the test refuses such data with an error that names the groups; before, rounding made the statistic a finite number (133.74, p = 7.7e-19, for a group of three observations of four variables beside two species of `iris`), a group of one gave `NA` and a single group `NaN`. With more observations than variables a group's covariance matrix is still singular when a variable is constant in the group, or a linear combination of the others in it; the test refuses that too, naming the group and saying which, where it returned `Inf` (p = 0) or `NaN` with warnings. A group is checked by the rank of its centred data at the tolerance of `qr()`. A missing value in `data` or in `group` still gives `NA`.

## Value

An object of class `htest`: `statistic`, named `M`, is Box's corrected statistic $M(1 - c_1)$; `parameter`, named `df`, its degrees of freedom $f_1$; `p.value` the upper tail of the chi-square distribution; and `method` and `data.name`.

## Size

In `inst/validation/box-m-approximations-size.R`, at 5000 replications per cell and the 5% level, with $g$ = 2, 3 or 5 groups of 10 to 100 observations of $p$ = 1 to 10 variables, the test held its level under Gaussian data in 60 of 67 cells, inside the release gate's interval, 0.042 to 0.058.

- **It is liberal when the groups are small for the number of variables.** With groups of 10 and 4 or 6 variables it rejected a true null hypothesis 0.055 to 0.106 of the time, outside the gate in four of those six cells (0.062 to 0.106, the most with five groups of 10 and six variables), and with groups of 20 and 10 variables 0.071 to 0.087. Box's F approximation, which the package does not compute, was closer to the level in each of those nine cells (0.051 to 0.070); in the other Gaussian cells the two differed by at most 0.005.
- **It is not robust to heavy tails.** With independent $t_5$ coordinates, three groups of 20, 50 or 100 and two or four variables, it rejected 0.26 to 0.48 of the time (0.18 to 0.46 with the factor of 0.15.0), more with larger groups: the test then detects the tails as well as unequal covariance matrices.
- **It is conservative with light tails.** On the data of the full sweep (`inst/validation/full-sweep-size-power.R`), 150 observations of `(y, x)` with `x` uniform and grouped by `cut(x, 3)`, it rejected 0.017 of the time, as Box's F did. The sweep's 0.012, at 400 replications, was measured with the factor of 0.15.0, which gives 0.016 in that design at 5000 replications: the low size comes from the uniform, light-tailed data, not from the approximation.

When the groups are small for the number of variables, or the data may have heavy tails, a rejection is weak evidence of unequal covariance matrices.

## Validation

`tests/testthat/test-boxm.R` checks the statistic against Box's approximation written out for two to five groups and one to four variables, and against the values of `biotools::boxM()` 4.3 and `heplots::boxM()` 1.8.6 on `iris` and on `mtcars`, and that two groups give the values of 0.15.0. `inst/validation/box-m-approximations-size.R` compares the chi-square and the F approximation with `biotools`, `heplots` and `DFA.CANCOR` on ten data sets, and measures the sizes above.

## References

Box, G. E. P. (1949). A general distribution theory for a class of likelihood criteria. *Biometrika*, 36(3/4), 317–346. doi:10.1093/biomet/36.34.317

## Examples

```
data(iris)
performBoxMTest(iris[,1:4], iris$Species)

# Two species: with two groups the values are those of 0.15.0
two <- droplevels(subset(iris, Species != "virginica"))
performBoxMTest(two[, 1:4], two$Species)
```

---

# performBPRandomEffectsTest

*Breusch-Pagan LM test for random effects*

---

## Description

Lagrange Multiplier test for the presence of a random individual effect in panel data. This is not a test for heteroscedasticity and does not respond to one; use the auxiliary-regression diagnostics for that.

## Usage

```
performBPRandomEffectsTest(model, data = NULL, id)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`, the pooled `lm()` of the panel; its observations may be in any order. |
| `data` | The data frame `model` was fitted to, holding the individual identifier named by `id`, or `NULL`, the default. The residuals and weights are read from the fit itself, and `id` from the rows of `data` that hold the observations of the fit. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`id`, not `factor(id)`); an identifier the model does not use needs `data`. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `id` | individual identifier column. |

## Details

The statistic is Breusch and Pagan (1980) equation 5, $LM = nT/(2(T-1))[\sum_i(\sum_t e_{it})^2/\sum_{it} e_{it}^2 - 1]^2$, which follows a chi-square distribution with one degree of freedom under the null of no individual effect. The bracketed ratio is close to one under the null and the statistic measures its squared departure from one. Before 0.11.0 the "- 1" and the square were absent and the scaling used T^2 rather than nT, which left the statistic sitting at the critical value: it rejected about a third of the time when no individual effect was present.

## Value

An object of class `htest`.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. The model is a pooled `lm` of the panel, with the units and the periods as columns of `data`; a `plm::plm()` fit is not an `lm` and is refused. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Breusch, T. S., & Pagan, A. R. (1980). The Lagrange Multiplier Test and Its Applications to Model Specification in Econometrics. *Review of Economic Studies*, 47(1), 239–253.

## Examples

```
df <- data.frame(id = rep(1:5, each = 4), time = rep(1:4, 5), x = runif(20), y = rnorm(20))
m <- lm(y ~ x, data = df)
performBPRandomEffectsTest(m, df, "id")
```

---

`performBPTest` *Perform Breusch-Pagan test for heteroscedasticity*

---

## Description

Implements the classical Breusch-Pagan (1979) test on a fitted linear model, in which the scaled squared residuals are regressed on the original regressors.

## Usage

```
performBPTest(model, data = NULL)
performBreuschPaganTest(model, data = NULL)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an `offset()` term, or takes its `weights` or `offset` from a column of the data needs no `data`, and the model frame of `model` is accepted as `data` (since 0.16.0; before, such models needed `data`). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |

## Details

The test statistic is half the explained sum of squares from regressing the scaled squared residuals $e_i^2/\hat{\sigma}^2 - 1$ (with $\hat{\sigma}^2 = \sum e_i^2/n$) on the explanatory variables. Under the null hypothesis of homoscedasticity *and normal disturbances* it follows a chi-square distribution with degrees of freedom equal to the number of regressors, matching `lmtest::bptest(..., studentize = FALSE)`. For the studentized $nR^2$ form that drops the normality assumption use `performKoenkerTest`.

The test is computed in memory whatever the size of the data. Up to 0.15.0 it switched to `performBPTestStreaming()` on data above 100 MB; call that function to stream.

## Value

An object of class `htest` containing the test statistic, p-value and degrees of freedom.

## Fixed effects for small groups

The Breusch-Pagan test, like White's and Koenker's tests, is not valid with fixed effects for small groups: a factor with many levels and few observations of each, such as `subject` in `lm(y ~ x + subject)` with each subject observed a few times. On homoscedastic data, with normal errors, it rejects too often, and the more often the more levels the factor has and the fewer observations each level has. The excess comes from the dummies of the factor in the auxiliary regression. The residuals of a level are taken from its own mean, so under normal errors the squared residuals of a level of $T$ observations vary between levels $T/(T-1)$ times as much as the auxiliary regression assumes, and with $G$ levels the statistic exceeds its degrees of freedom by about $(G-1)/(T-1)$, as Koenker's did in the study below. Regressed on `x` alone, without the dummies, the squared residuals were rejected by this test in 4.3% to 5.6% of the draws (`inst/validation/fixed-effects-dummies.R`). The excess grows with $G$ at a fixed $T$, while the spread of the chi-squared distribution grows as $\sqrt{2G}$, so with a fixed number of observations per level the size goes to 1 as the number of levels grows.

Since 0.16.0 the test warns when a term of the model that holds a factor or a character variable has more levels than observations per level: $L > n/L$, that is $L^2 > n$, with $L$ the number of distinct combinations of the factors of the term among the $n$ observations of the fit. The term is a factor, such as `subject`; an interaction of factors, counted by its cells, such as `a:b` of a crossed `a * b` or `school:class` of a nested `school / class`; or a slope for each level, `x:subject`. A variable left

out of the formula, as in `y ~ . - id`, does not count. The warning names the term and gives $L$, $n$ and the mean number of observations per level; the statistic, the degrees of freedom and the p-value are those the test computes without it. It has class `heteroTests_fixed_effects_warning`, which `suppressWarnings(expr, classes = "heteroTests_fixed_effects_warning")` muffles, and no other warning. It is given on every call, also when `cachedTest()` or `runHeteroTests()` answers from its cache, and `runHeteroTests()` and `runDiagnostics()` give it once for each test that gives it. `performBPTestStreaming()`, `performBPTestRobust()` with `studentize = FALSE`, the deprecated `performBreuschPaganTest()` and the registered `"breusch_pagan"` give it too.

On a balanced design of $G$ subjects of $T$ observations the rule is $G \leq T$. In the study it kept the three tests within 2.5% to 7.5% at the 5% level where it holds, and put at least one of them outside where it does not, in 68 of the 69 designs where they computed; the exception is this test on 20 subjects of 20, at 8.0%. Unbalanced groups, a second regressor and a regressor correlated with the subject effects followed the same pattern, with the mean number of observations per level, $n/G$, in place of $T$. The rule is a warning, not a correction, and many panels fail it: on 100 firms observed over 10 years the three tests reject in 18% to 21% of the draws. Rejection at the 5% level, in percent, in 2,000 homoscedastic draws of `lm(y ~ x + subject)`, with `x`, the subject effects and the errors standard normal:

| Subjects | Observations of each | White | Koenker | Breusch-Pagan | Warns |
|---|---|---|---|---|---|
| 10 | 10 | 6.4 | 6.2 | 7.3 | no |
| 20 | 20 | 6.0 | 6.4 | 8.0 | no |
| 5 | 4 | 1.0 | 5.4 | 7.0 | yes |
| 30 | 3 | 50.7 | 57.3 | 39.6 | yes |
| 30 | 4 | 22.2 | 23.9 | 28.7 | yes |
| 100 | 8 | 23.8 | 23.8 | 25.2 | yes |
| 100 | 10 | 18.1 | 18.4 | 21.3 | yes |
| 200 | 4 | 97.8 | 97.6 | 82.5 | yes |

A slope for each level counts too, but there the error goes the other way: with a slope for each subject, `lm(y ~ x + x:subject)`, on 30 subjects of 3 and of 4 and on 100 of 8, the three tests rejected too rarely, White's in 0.7% to 2.2% of the draws, Koenker's in 0.1% to 0.8% and the Breusch-Pagan test in 0.4% to 2.8% (`inst/validation/fixed-effects-variants-size.R`). The warning therefore says that the p-value is unreliable: in the simulations usually far too small, and too large with a slope per level.

The study is `inst/validation/fixed-effects-size.R`, with its results in `fixed-effects-size.csv` (`system.file("validation", package = "heteroTests")`). Juhl and Sosa-Escudero (2014) derive a test of heteroskedasticity for this setting, after fixed effects estimation of a panel of many units observed a fixed number of times; it needs to be told which factor is the unit, and the package does not implement it.

## Deprecated name

`performBreuschPaganTest()` is deprecated since 0.14.0. It was the same function as `performBPTest()` under a second name; it still works, returns what `performBPTest()` returns and warns. Use `performBPTest()`.

### Weighted fits

On a fit with weights the test is computed from the Pearson residuals $\sqrt{w_i}\, e_i$, so it tests whether the weights are adequate; see the section on weighted fits in `performKoenkerTest()`, which also covers weights estimated by `fitWLS()`.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Breusch, T. S., & 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:10.1016/03044076(81)900622

Juhl, T., & Sosa-Escudero, W. (2014). Testing for heteroskedasticity in fixed effects models. *Journal of Econometrics*, 178(Part 3), 484–494. doi:10.1016/j.jeconom.2013.07.005

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performBPTest(m, mtcars)
```

---

## `performBPTestRobust` *Robust Breusch–Pagan test with bootstrap and effect sizes*

---

### Description

Enhances `performBPTest()` by optionally studentising residuals, resampling the test statistic via bootstrap, and reporting effect sizes together with power diagnostics.

### Usage

```
performBPTestRobust(
  model,
  data = NULL,
  studentize = TRUE,
  bootstrap = FALSE,
  B = 999,
  ci_level = 0.95,
  parallel = FALSE,
```

```
  studentized
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object describing the mean structure whose residual variance is to be assessed. |
| data | The data frame model was fitted to, or NULL, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.17.0; before, such models needed data). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| studentize | Logical, use studentized residuals for the auxiliary regression (Koenker variant)? Defaults to TRUE. The name matches lmtest::bptest() and performHarveyTest(). |
| bootstrap | Logical, compute bootstrap diagnostics for the statistic and p-value? The bootstrap is bootstrapTestStatistic(), so on a fit returned by fitWLS() each replicate estimates the weights again, its default since 0.15.0. There is no argument to hold them fixed here; bootstrapTestStatistic() with reestimate_weights = FALSE does that. |
| B | Number of bootstrap replications when bootstrap = TRUE. Defaults to 999, at which a bootstrap p-value of the form (1 + #) / (B + 1) gives an exact test at the 5% and 1% levels; it was 1000 before 0.15.0. The value is recorded in robust_details$bootstrap$B whether or not the bootstrap runs. |
| ci_level | Confidence level for reported intervals. |
| parallel | Logical, allow parallel bootstrap evaluation when the parallel package is available. A seeded call gives the same result each time; see *Parallel replicates* in bootstrapTestStatistic(). |
| studentized | Deprecated since 0.15.0: use studentize. It is still accepted, with a warning, and does what studentize does; it stops working in 1.0.0. |

## Details

Offers expanded reporting for the Breusch–Pagan family of tests including bootstrap p-values and asymptotic confidence intervals. The optional studentize argument toggles between the classic and Koenker versions.

## Value

An augmented htest object containing a robust_details element describing the additional diagnostics. Its studentized field records the value of studentize.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It accounts for an offset with `studentize = FALSE`, and since 0.16.0 with `studentize = TRUE`, the default, whose refit of the design carries the offset of the fit. It refuses a multivariate `lm` ("needs a model with one response"), and, since 0.16.0, with `studentize = TRUE`, the default, an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, the basis of its smooths, that the studentized statistic fits again, and, since 0.17.0, such a fit with `bootstrap = TRUE`, whose replicates are fitted again. With `studentize = FALSE` and no bootstrap it tests such a fit on its design, as `performBPTest()` does. Since 0.17.0 a `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is tested on that design and gives the value of the `lm()` of it; before, it was refused as a penalised one. A `gam` without smooth terms is tested as the Gaussian `glm` it is (decisions 37 and 38 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Breusch, T. S., & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica, 47*(5), 1287–1294.

Koenker, R. (1981). A note on studentizing a test for heteroscedasticity. *Journal of Econometrics, 17*(1), 107–112.

### See Also

`performBPTest()` for the baseline test and `performKoenkerTest()` for the standalone studentised variant.

### Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
performBPTestRobust(mod, mtcars, bootstrap = TRUE, B = 200)
```

---

## `performBPTestStreaming` *Streaming Breusch-Pagan test for large datasets*

---

### Description

Computes the classical Breusch-Pagan statistic chunk by chunk so that large datasets can be evaluated without materialising the full auxiliary regression in memory. The streamed result is algebraically identical to `performBPTest`.

## Usage

```
performBPTestStreaming(model, data = NULL, chunk_size = 10000, progress = interactive())
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object describing the mean structure whose residual variance is to be assessed. |
| data | The data frame model was fitted to, or NULL, the default, read as performWhiteTestStreaming() reads it: the response, design, weights and offset come from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data (since 0.16.0), and data that are given are matched to the observations of the fit. |
| chunk_size | Positive integer giving the number of observations processed per streaming chunk. Smaller values reduce peak memory usage at the expense of additional iteration overhead. |
| progress | Logical flag controlling whether a textual progress bar is displayed while chunks are processed. Defaults to interactive(). |

## Details

Each chunk of the regressors, centred at their means, is reduced by a QR decomposition stacked on the triangular factor of the chunks before it, as in performWhiteTestStreaming(), since 0.17.0. Up to 0.16.0 the cross-products of the regressors were accumulated: those of a regressor 1,000 standard deviations from zero looked collinear with the intercept, and the test stopped ("Breusch-Pagan auxiliary regression became rank deficient"); and squared residuals of about $10^{-10}$ were taken for constant ("requires variation in squared residuals"). It gives the value of performBPTest() on both. Sparse cross-products from **Matrix** are no longer used.

Since 0.16.0 the chunks are rows of the design of the fit, as in performWhiteTestStreaming(), whose details say what changed.

## Value

A htest object mirroring performBPTest and reporting the chi-squared statistic, degrees of freedom, p-value, and metadata describing the chunked computation.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, data that lack rows or variables of the fit, and, since 0.16.0, a multivariate lm, a fit changed after it was fitted and a fit with model = FALSE whose data are gone, unless data gives its response and fitted values. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## See Also

performBPTest for the standard implementation.

## Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
performBPTestStreaming(mod, mtcars, chunk_size = 16, progress = FALSE)
```

---

# performBrownForsytheTest

*Perform Brown-Forsythe test for equality of variances*

---

## Description

Brown-Forsythe test using medians instead of means.

## Usage

```
performBrownForsytheTest(model, data = NULL, group)
```

## Arguments

| | |
|---|---|
| model | an object of class lm. |
| data | The data frame model was fitted to, or NULL, the default. The residuals and weights are read from the fit itself, and group from the rows of data that hold the observations of the fit; the observations whose group is missing are left out, with a warning. Without data it is read from the model frame of model, which holds it only when it is a variable of the model under its own name (x, not log(x)); a grouping variable the model does not use needs data. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| group | name of the grouping variable. |

## Details

The absolute deviations from the group medians are compared across groups. The resulting statistic is an $F$ ratio with $k - 1$ and $N - k$ degrees of freedom.

## Value

An object of class htest containing the F statistic, p-value and degrees of freedom.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An `mgcv::gam()` fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context); since 0.16.0 its observations are matched to the rows of `data` by the row names of its model frame. Before, they were matched by position, since the residuals of a `gam` carry no names, and `data` with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Brown, M. B., & Forsythe, A. B. (1974). Robust tests for the equality of variances. *Journal of the American Statistical Association*, 69(346), 364–367.

### Examples

```
data(mtcars)
mtcars$cyl <- factor(mtcars$cyl)
m <- lm(mpg ~ wt, data = mtcars)
performBrownForsytheTest(m, mtcars, "cyl")
```

---

## performCookWeisbergTest

*Perform Cook-Weisberg test for heteroscedasticity*

---

### Description

The Cook-Weisberg (1983) score test, with the fitted values as the sole variance regressor by default, or against a variance model given by `var_formula`. Since 0.14.0 it is the one entry point for the test; `performNCVTest` is deprecated.

### Usage

```
performCookWeisbergTest(model, data = NULL, var_formula = NULL)
```

### Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | optional data frame the model was fitted to, in which `var_formula` is evaluated. When `NULL`, the default, the model frame of `model` is used. Without `var_formula` the statistic is computed from the residuals and fitted values of `model`, and `data` does not change the result. With it, the variables of |

var_formula are read at the rows of data that hold the observations of the fit, matched as the tests that read data match them: data with row names of their own in any order, data that number their rows (a tibble, a data.table) in the order of the fit. Since 0.16.0 that includes numbered data with one row per observation, such as the complete rows of a fit that dropped rows numbered again, unless two rows of the fit tie on every column of the model frame and var_formula names a variable beyond it. A variable of var_formula is read from data, or without data from the model frame, which holds the variables of the model under their own names; since 0.16.0 one that neither holds is refused, where it was found in the workspace, whatever the order of its values.

var_formula optional one-sided formula giving the variance model, evaluated in data (for example ~ x1 + x2). When NULL, the default, the fitted values are the variance regressor and the result is what this function returned before 0.14.0; otherwise it is what the deprecated performNCVTest(model, var_formula) returned, labels included. In 0.14.0 var_formula was the second argument: a formula passed there by position is still read as var_formula, with a warning that shows the new call; that form stops working in 1.0.0.

## Details

Writing $\hat{\sigma}^2 = \sum_i \hat{e}_i^2/n$ for the maximum-likelihood error variance, the test regresses the scaled squared residuals $\hat{e}_i^2/\hat{\sigma}^2$ on the fitted values and refers half the explained sum of squares to a chi-square distribution with one degree of freedom. This is the diagnostic returned by Stata's estat hettest with the fitted option, and it is exactly performNCVTest with its default variance model.

The chi-square reference distribution follows from the null variance of $\hat{e}^2/\sigma^2$ being 2, which holds under normal errors. When that assumption is doubtful, prefer performKoenkerTest.

## Value

An object of class htest containing the score statistic, its single degree of freedom and the p-value.

## Validation

Reproduces car::ncvTest() to within 1e-8; see 'tests/testthat/test-pass-a-reference.R'. Releases before 0.7.0 returned $nR^2$ from regressing the raw squared residuals on the fitted values, which is the studentized (Koenker) statistic rather than the Cook-Weisberg score test.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, and data that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate lm ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with model = FALSE whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without data. A fit with model = FALSE whose data are gone is tested on data that give its response and fitted values, and refused without them. Since 0.16.0 data

is matched to the observations of the fit as by every test, with or without `var_formula`: without it the test reads nothing from `data`, and gave the value of the fit on numbered data in another order, rows in front of those of the fit and repeated row names, which are refused. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Cook, R. D., & Weisberg, S. (1983). Diagnostics for heteroscedasticity in regression. *Biometrika*, 70(1), 1–10. doi:10.1093/biomet/70.1.1

## See Also

`performNCVTest`, `performKoenkerTest`, `performBPTest`

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performCookWeisbergTest(m)

# Score test against a specific variance model rather than the fitted values
performCookWeisbergTest(m, var_formula = ~wt)

# With the data, the variance model can use variables the mean model does not
performCookWeisbergTest(m, mtcars, var_formula = ~ wt + hp)
```

---

# performDavidianCarrollTest

*Perform Davidian-Carroll test*

---

## Description

Regresses `log(residual^2)` on fitted values using a polynomial.

## Usage

```
performDavidianCarrollTest(model, data = NULL, degree = 2)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | optional data frame the model was fitted to. When `NULL`, the default, the model frame of `model` is used. The statistic is computed from the residuals and fitted values of `model`, so `data` does not change the result; it is accepted so that every test takes `(model, data, ...)`. |
| `degree` | polynomial degree. Before 0.15.0 `degree` was the second argument: a number passed there by position is still read as `degree`, with a warning that shows the new call; that form stops working in 1.0.0. |

## Details

A polynomial in the fitted values is fitted to $\log(e^2)$. Joint significance of the polynomial terms is evaluated with an $F$ statistic.

## Value

An object of class `htest`.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate `lm` ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without `data`. A fit with `model = FALSE` whose data are gone is tested on `data` that give its response and fitted values, and refused without them. The test reads nothing from `data`, and since 0.16.0 matches it to the observations of the fit as every test does: data with row names of their own in any order, data that number their rows (a tibble, a data.table) in the order of the fit; other numbered data, rows in front of those of the fit and repeated row names are refused, where it gave the value of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Davidian, M., & Carroll, R. J. (1987). Variance function estimation. *Journal of the American Statistical Association*, 82(400), 1079–1091.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performDavidianCarrollTest(m)
```

---

`performFlignerKilleenTest` *Perform Fligner-Killeen test for homogeneity of variances*

---

## Description

The median-centred Fligner-Killeen test of homogeneity of variances, on the Pearson residuals of the fit, computed with `fligner.test()`: it ranks the absolute deviations of the residuals from their group medians and scores the ranks by quantiles of the normal distribution, which makes it robust to outliers and to errors that are not normal.

## Usage

```
performFlignerKilleenTest(model, data = NULL, group)
```

## Arguments

| | |
|---|---|
| model | an object of class lm. |
| data | The data frame model was fitted to, or NULL, the default. The residuals and weights are read from the fit itself, and group from the rows of data that hold the observations of the fit; the observations whose group is missing are left out, with a warning. Without data it is read from the model frame of model, which holds it only when it is a variable of the model under its own name (x, not log(x)); a grouping variable the model does not use needs data. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| group | name of the grouping variable. |

## Details

The test is the median-centred version of the test of Fligner and Killeen (1976), the "F-K:med $X^2$" of Conover, Johnson and Johnson (1981), whose simulation study found it among the tests of homogeneity of variances most robust to departures from normality; it is the test fligner.test() computes. The residuals of each group are centred at the median of the group, the absolute values of the centred residuals are ranked over all $n$ observations, and the rank $r_i$ is given the score $a_i = \Phi^{-1}\{(1 + r_i/(n+1))/2\}$. With $\bar{a}_j$ the mean score of group $j$, of size $n_j$, $\bar{a}$ the mean of all the scores and $s_a^2 = \sum_i (a_i - \bar{a})^2/(n-1)$, the statistic

$$X^2 = \sum_j n_j(\bar{a}_j - \bar{a})^2/s_a^2$$

is referred to a chi-squared distribution with $k-1$ degrees of freedom, for $k$ groups. No logarithm is taken. The observations whose group is missing are left out before the statistic is computed.

Tied values are given the mean of their ranks. Since 0.16.0 the deviations are those of the residuals $r_i = \sqrt{w_i}\,(y_i - x_i^\top \hat{\beta} - o_i)$ computed observation by observation, as the Gaussian glm computes them, and not the residuals an lm stores, which its QR decomposition rounds differently from one observation to the next. For an mgcv::gam() fit $x_i$ is the row of its linear predictor matrix. Observations with the same values of the model's variables, weight and offset, such as repeated values of an integer response or of a response recorded with few digits, are then given the same residual, as in exact arithmetic, where their residuals agree to within rounding, even when a basis such as poly() gives them design rows that differ in their last bits; a term of the row position, such as seq_along(x), keeps them apart. This needs the model's variables: it is done when data is given, or when the model frame holds them, as it holds a variable the model uses under its own name (x, not poly(x, 3)). And absolute deviations closer than $8\epsilon s_j$, with $\epsilon$ = .Machine$double.eps and $s_j$ the largest $|r_i|$ of group $j$ (the smaller of the two for deviations of different groups), are ranked as tied: in a group of even size the two residuals in the middle lie at the same distance from

the median of the group, half their difference, a tie that the centring breaks by up to $\epsilon s_j$. On data with repeated observations the statistic can differ much from before: on `lm(extra ~ group, data = sleep)` it was 0.154 and is 0.213, the value of exact ties. On continuous data it moves by much less (see `NEWS`). Where neither changes a rank, the value is that of `fligner.test()` on the residuals of the fit, as before. Two limits remain. A tie between different observations that exact decimal arithmetic would give, as when two pairs of observations of a regressor recorded with few digits differ by the same amounts, is restored only when rounding leaves it within the tolerance, and two forms of one model can then differ by several percent: on 70 observations that repeat 30 values of $x$ recorded with one decimal, `y ~ poly(x, 3) + g` gives 1.21547, and `y ~ x + I(x^2) + I(x^3) + g` and `y ~ poly(x, 3, raw = TRUE) + g` give 1.30280, since `poly()` breaks in its last bits the ties between repeated values of $x$ in different groups. And in a group whose residuals are far from zero, $10^8$ times their spread, the tolerance may also tie two deviations that differ by a few units in the last place of the residuals.

## Value

An object of class `htest` containing the chi-squared statistic, p-value and degrees of freedom.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An `mgcv::gam()` fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context); since 0.16.0 its observations are matched to the rows of `data` by the row names of its model frame. Before, they were matched by position, since the residuals of a `gam` carry no names, and `data` with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Conover, W. J., Johnson, M. E., & Johnson, M. M. (1981). A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data. *Technometrics*, 23(4), 351–361. doi:10.1080/00401706.1981.10487680

Fligner, M. A., & Killeen, T. J. (1976). Distribution-free two-sample tests for scale. *Journal of the American Statistical Association*, 71(353), 210–213. doi:10.1080/01621459.1976.10481517

## Examples

```
data(mtcars)
mtcars$cyl <- factor(mtcars$cyl)
m <- lm(mpg ~ wt, data = mtcars)
performFlignerKilleenTest(m, mtcars, "cyl")
```

performGlejserTest *Perform Glejser test for heteroscedasticity*

## Description

Regresses absolute residuals on a transformation of a suspected variable. By default, `robust = TRUE` since 0.15.0, the absolute residuals are first corrected for the asymmetry of the errors: this is the version of Im (2000) and of Machado and Santos Silva (2000), which keeps its level when the errors are asymmetric, from about 150 observations on (see the section on asymmetric errors for smaller samples). `robust = FALSE` gives Glejser's (1969) statistic, the default up to 0.14.0.

## Usage

```
performGlejserTest(model, data = NULL, variable,
                   transformation = c("abs", "sqrt", "inverse", "inverse_sqrt"),
                    robust = TRUE)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The residuals and weights are read from the fit itself, and `variable` from the rows of `data` that hold the observations of the fit. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`x`, not `log(x)`); a model that transforms the variable needs `data`, as does a variable the model does not use. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `variable` | name of the suspected variable. |
| `transformation` | transformation applied to `variable`. |
| `robust` | logical. When `TRUE`, the default since 0.15.0, the absolute residuals are corrected for the asymmetry of the errors before the auxiliary regression. When `FALSE` the statistic is Glejser's (1969), which has its nominal level when positive and negative errors are equally likely, as they are for symmetric errors; it is what the function returned by default up to 0.14.0. See the sections on asymmetric errors and on the default. |

## Details

A variety of transformations of the explanatory variable (e.g. absolute value, square root) can reveal a relationship between scale and the covariate. Significance of the slope parameter is assessed with a t test. For a weighted fit the residuals are the Pearson residuals.

Glejser's statistic, `robust = FALSE`, has its nominal level only when positive and negative errors are equally likely. The corrected statistic of the default does not need that, and is the one to use when the residuals may be skewed or their shape is unknown.

Either form conditions on one variable and one transformation. Scanning several transformations and reporting only the smallest p-value invalidates the nominal level. Simulated size and power of both statistics are recorded in `inst/validation/glejser-skewed-errors.csv`; those of Glejser's statistic in `inst/validation/pass-a-size-power.csv`, and those of the default in `inst/validation/full-sweep-size-power.csv` and `inst/validation/weighted-fits-size.csv`.

## Value

An object of class `htest` containing the t statistic, p-value and degrees of freedom. The `method` element says which of the two statistics was computed.

## Asymmetric errors

Glejser's statistic takes the absolute residuals for the absolute errors. The two differ by the estimation error of the mean equation,

$$|\hat{e}_i| \approx |e_i| - \operatorname{sign}(e_i)\, x_i^{\top}(\hat{\beta} - \beta),$$

and the second term averages out only when positive and negative errors are equally likely (Godfrey 1996). When they are not, and the variable tested is correlated with the regressors, the test does not in general have its nominal level, and a larger sample does not help. In the designs of `inst/validation/glejser-skewed-errors.R` Glejser's statistic, `robust = FALSE`, rejects a true null hypothesis about 13% of the time at the 5% level under exponential errors and about 17% under lognormal errors, at every sample size from 50 to 1000. For other error distributions the rate can also fall below the nominal level.

`robust = TRUE` replaces the regressand $|\hat{e}_i|$ by

$$|\hat{e}_i| - \hat{m}\,\hat{e}_i, \qquad \hat{m} = \frac{1}{n}\sum_{i=1}^{n} \operatorname{sign}(\hat{e}_i),$$

where $\hat{m}$ is the proportion of positive residuals less the proportion of negative ones. This is the correction of Im (2000). The regressand of Machado and Santos Silva (2000), $\hat{e}_i\,[I(\hat{e}_i \geq 0) - \hat{\eta}]$ with $\hat{\eta}$ the proportion of non-negative residuals, is half of it when no residual is exactly zero, and the two then give the same test. A residual that is zero up to rounding error is counted as zero, so that the statistic does not depend on the sign of a rounding error.

The auxiliary regression is otherwise the same. Both references state the statistic as $nR^2$ of that regression, referred to a chi-squared distribution with one degree of freedom. The t statistic reported here is the signed form of it, $nR^2 = nt^2/(t^2+n-2)$, and its p-value is taken from the t distribution with $n-2$ degrees of freedom, as for Glejser's statistic. The two p-values agree as the sample grows.

In the same study, with 5000 replications in each cell, the corrected statistic is within simulation error of the 5% level from 150 observations on: all 27 of its cells with a true null hypothesis and 150 observations or more are inside the 99% interval around 5%, from 4.21% to 5.79%, at 4.4% to 5.7%. At 50 observations it rejects 5.7% to 6.6% of the time under skewed errors, and four of those six cells are above that interval, at 5.80% to 6.58%. Under symmetric errors $\hat{m}$ is close to zero and the two statistics differ little: the corrected one rejects 0.5 to 0.7 percentage points less often at 50 observations and gives up one or two points of power there. For a weighted fit the correction is applied to the Pearson residuals.

### The default

Up to 0.14.0 the default was `robust = FALSE`. Since 0.15.0 it is `robust = TRUE`, so a call that does not give `robust` returns a different statistic and p-value from those it returned before, and its `method` ends in "(robust to asymmetric errors)". `robust = FALSE` returns what such a call returned up to 0.14.0.

Glejser's statistic needs positive and negative errors to be equally likely, which is seldom known: in the study above it rejects a true null hypothesis 8.8% to 17.7% of the time at the 5% level when the errors are skewed, at every sample size from 50 to 1000, where the corrected statistic rejects 4.6% to 5.7% of the time from 150 observations on. The change costs some power when the errors are symmetric, 1.1 and 1.9 percentage points at 50 observations under Gaussian errors in that study, and the corrected statistic is above the 5% level at 50 observations under skewed errors, as above.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An `mgcv::gam()` fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context); since 0.16.0 its observations are matched to the rows of `data` by the row names of its model frame. Before, they were matched by position, since the residuals of a `gam` carry no names, and `data` with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Glejser, H. (1969). A new test for heteroskedasticity. *Journal of the American Statistical Association*, 64(325), 316–323.

Godfrey, L. G. (1996). Some results on the Glejser and Koenker tests for heteroskedasticity. *Journal of Econometrics*, 72(1–2), 275–299. doi:10.1016/03044076(94)017220

Im, K. S. (2000). Robustifying Glejser test of heteroskedasticity. *Journal of Econometrics*, 97(1), 179–188. doi:10.1016/S03044076(99)000615

Machado, J. A. F. and Santos Silva, J. M. C. (2000). Glejser's test revisited. *Journal of Econometrics*, 97(1), 189–202. doi:10.1016/S03044076(00)000166

### See Also

`performKoenkerTest` for a test on the squared residuals that needs neither normal nor symmetric errors; `performParkTest` and `performHarveyTest` for related parametric tests.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
# The default, which keeps its level under asymmetric errors from about
```

```
# 150 observations
performGlejserTest(m, mtcars, "wt")

# Glejser's (1969) statistic, the default up to 0.14.0
performGlejserTest(m, mtcars, "wt", robust = FALSE)

# Examine an alternative transformation
performGlejserTest(m, mtcars, "wt", transformation = "inverse")
```

---

`performGQTest` *Perform Goldfeld-Quandt test for heteroscedasticity*

---

## Description

Implements the Goldfeld-Quandt test on a fitted linear model with directional or two-sided alternatives.

## Usage

```
performGQTest(model, data = NULL, order_by, fraction = 0.2,
  alternative = c("greater", "two.sided", "less"))
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | the data frame `model` was fitted to, or `NULL`, the default. The response, design, weights and offset are read from the fit itself, and `order_by` from the rows of `data` that hold the observations of the fit. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`x`, not `log(x)`); a model that transforms the variable it is ordered by needs `data`. Since 0.16.0 the two parts are fitted to their rows of the stored design, with the offset of the fit; before, an unweighted fit was refitted from its formula on each part of `data`, which computed again a term that depends on the rows it is computed from, such as `ns()` or `I(x > median(x))`, and an offset was left out: an `offset =` argument of an unweighted fit, any offset of a weighted one. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `order_by` | single column name used to order observations. |
| `fraction` | fraction of observations omitted from the middle; must lie strictly between zero and one. |
| `alternative` | alternative hypothesis: `"greater"` for increasing variance from segment 1 to segment 2, `"less"` for decreasing variance, or `"two.sided"` for either direction. |

## Details

Observations are ordered by a suspected variance-driving variable. With the split point fixed at the sample midpoint, a central fraction is omitted and the model is fitted again on the lower and upper segments: each segment is fitted to its rows of the design, the response and the offset of the fit, as `lmtest::gqtest()` fits them to the rows of the design. The statistic is the residual mean square of segment 2 divided by the residual mean square of segment 1 and is therefore directional rather than being forced above one. Split arithmetic and p-value conventions match `lmtest::gqtest(..., point = 0.5)`.

Before 0.16.0 the segments of an unweighted fit were refitted from the formula, which computed again, on the rows of each segment, a term that depends on the rows it is computed from: `ns(x)` and `bs(x)` placed their knots at quantiles of the segment, and `I(x > median(x))` took its median, so the segments were fitted to columns the fit never used; `poly(x)` computed another basis of the same columns, which changed the last bits only. The refit also left out an offset given as an argument, and a weighted fit was refitted without its offset.

## Value

An object of class `htest` containing the directional GQ statistic, p-value, degrees of freedom and alternative hypothesis.

## Weighted fits

On a fit with weights each segment is refitted with its own weights and the segments are compared on the weighted residual sum of squares, $\sum w_i e_i^2$. That is the Goldfeld-Quandt statistic of the equivalent unweighted regression, so the test asks whether the weights are adequate. Releases before 0.12.0 refitted the segments without the weights.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It accepts a model with an offset, given as an argument or as an `offset()` term, weighted or not, and gives the value of the model fitted to the response minus the offset (since 0.16.0). It refuses a multivariate `lm` ("needs a model with one response"), and, since 0.16.0, an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, the basis of its smooths, that the test fits again. Since 0.17.0 a `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is tested on that design and gives the value of the `lm()` of it; before, it was refused as a penalised one. A `gam` without smooth terms is tested as the Gaussian `glm` it is (decisions 37 and 38 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Goldfeld, S. M., & Quandt, R. E. (1965). Some tests for homoscedasticity. *Journal of the American Statistical Association*, 60(310), 539–547. doi:10.1080/01621459.1965.10480811

Greene, W. H. (2018). *Econometric Analysis* (8th ed.). Pearson.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performGQTest(m, mtcars, order_by = "wt")
performGQTest(m, mtcars, order_by = "wt", alternative = "two.sided")
```

---

`performHarrisonMcCabeTest` *Harrison-McCabe test for heteroscedasticity*

---

## Description

Orders the observations by a variable and tests whether the share of the residual sum of squares that falls in the first part of the ordered sample is too small, or too large, for errors of constant variance. The p-value is exact under normal errors, given the regressors.

## Usage

```
performHarrisonMcCabeTest(
  model,
  data = NULL,
  order_by,
  point = 0.5,
  alternative = c("less", "two.sided", "greater")
)
```

## Arguments

| | |
|---|---|
| `model` | A fitted stats::lm object, with or without weights, or a stats::glm with the Gaussian family and the identity link. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The residuals, design and weights are read from the fit itself, and so, since 0.16.0, is its rank, decided at the tolerance of the fit (see the section on the rank), where the test decided the rank of the design again at `1e-7`; `order_by` is read from the rows of `data` that hold the observations of the fit. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`x`, not `log(x)`); a model that transforms the variable it is ordered by needs `data`, as does an ordering variable the model does not use. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |

`order_by` Character scalar naming the column of `data` that orders the observations. Tied values keep the order the observations have in the fitted model.

`point` The size of the first part of the ordered sample. A number strictly between 0 and 1 is a fraction of the sample, and the first part then holds `floor(point * n)` observations, as in `lmtest::hmctest()`. A whole number of 1 or more is the number of observations. Each part must hold at least as many observations as the model has coefficients.

`alternative` Character scalar. `"less"` (the default) is the alternative that the variance increases with `order_by`, which makes the share small; `"greater"` that it decreases; `"two.sided"` either. The two-sided p-value is twice the smaller of the two one-sided ones, capped at 1, as in `performGQTest()` and in the default of `skedastic::harrison_mccabe()`.

## Details

With $\hat{e}_{(1)}, \ldots, \hat{e}_{(n)}$ the least-squares residuals in the order of `order_by`, the statistic of Harrison and McCabe (1979) is

$$b = \frac{\sum_{i=1}^{m} \hat{e}_{(i)}^2}{\sum_{i=1}^{n} \hat{e}_{(i)}^2},$$

the share of the residual sum of squares that falls in the first $m$ observations. A variance that increases along the ordering makes $b$ small, and one that decreases makes it large. It is the statistic of `lmtest::hmctest()`.

The 1979 article was not consulted for this implementation. The definition follows Harrison (1981), by its first author, which restates the statistic with $m$ the integer part of $n/2$ and rejects for small $b$, and it agrees with `lmtest::hmctest()`.

Under normal errors of constant variance, $b$ is a ratio of quadratic forms in normal variables, and its distribution depends on the regressors. With $M$ the residual-maker of the design in the ordered sample and $D$ the diagonal matrix that selects the first $m$ observations,

$$P(b \le b_0) = P\left(\sum_j (\lambda_j - b_0) z_j^2 \le 0\right),$$

where the $z_j$ are independent standard normal and the $\lambda_j$ are the $n - k$ eigenvalues of $MDM$ on the residual space, $k$ being the number of coefficients. They are found from the $k$ singular values of the first $m$ rows of an orthonormal basis of the design, and the probability is evaluated by Imhof's (1961) formula with `stats::integrate()`, as Harrison (1981) does for the exact variant of the test. For a constant and a linear trend, the 5% critical values this gives at 10, 20, 30 and 40 observations are those of Harrison's (1981) Table 3 to the three decimals printed there. Under the null hypothesis the mean of $b$ is $(m - \sum_{i \le m} h_{(i)})/(n - k)$, with $h_{(i)}$ the leverages in the ordered sample.

The probability is computed to an absolute accuracy of about `2e-11`. Smaller p-values are not resolved, and a statistic far in a tail can return a p-value of 0.

The p-value depends only on the space spanned by the regressors, so it does not change with the parametrisation of the model.

### Value

An object of class `htest` with the share `HMC` as the statistic, the number of observations in the first part `m` and the sample size `n` as parameters, the p-value, and the alternative in words.

### The rank

$k$ is the rank of the fit, the number of its coefficients that are not aliased, decided at the tolerance the fit used: `1e-7` for an `lm`, or the `tol` it was given, and `1e-11` for a `glm` (since 0.16.0). Before, the rank was decided again at `1e-7`, so a Gaussian `glm`, or an `lm` fitted with a smaller `tol`, whose design holds a column that is a combination of the others to within `1e-7` but not to within its own tolerance, was given a null distribution of a smaller rank than its residuals. With `x3 = x1 + 1e-9 * rnorm(120)` the `glm` has rank 4, and its p-value is 0.002273 where it was 0.002288, computed with rank 3. On every other fit the two ranks agree and the value is unchanged.

### Direction of the alternative

In `performGQTest()` and `performSzroeterTest()` the statistic grows when the variance increases along the ordering, and in this test it shrinks. The alternative names the tail of each statistic, so the same hypothesis, a variance that increases with `order_by`, is `alternative = "less"` here and `alternative = "greater"` in those two functions. In all three it is the default, and the `alternative` element of the result states it in words.

### Weighted fits

On a fit with weights the statistic is computed from the Pearson residuals $\sqrt{w_i}\,\hat{e}_i$, and the null distribution from the weighted design $\sqrt{w_i}\,x_i$. Both are those of the equivalent unweighted regression, so the test asks whether the weights are adequate. The weights of a Gaussian `glm` are its prior weights.

### Heavy tails

The null distribution assumes normal errors. In the designs of `inst/validation/harrison-mccabe-size-power.R` the test rejects a true null hypothesis 11.7% to 16.3% of the time at the 5% level when the errors follow a $t_5$ distribution, and more often the larger the sample; `performGQTest()` and `performSzroeterTest()` do the same. When the tails may be heavy, use `performGlejserTest()` with the ordering variable as `variable` and `robust = TRUE`, which in the same study rejects 4.7% to 4.9% of the time under those errors. Its default transformation takes the absolute value of the variable, which preserves the ordering only when the variable is positive.

### Reference implementations

The statistic agrees with `lmtest::hmctest()` to `1e-8`. Its p-value does not: with `simulate.p = TRUE` it is simulated from residuals of a regression on a constant alone, whatever the model, and with `simulate.p = FALSE` it is `NA`. `skedastic::harrison_mccabe()` reports a different p-value as well; `inst/validation/README.md` says why.

### Supported models

It accepts an `lm` and, since 0.16.0, a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on

the estimable part of its design: it gives the value of the same model fitted without the aliased columns. The Gaussian `glm` with prior weights `w` is the least-squares fit `lm(..., weights = w)`: it has the same design, residuals and weights, and it states the normal errors of constant variance, given the weights, under which the null distribution above is exact, so the test gives the value of the `lm` when the two decide the rank alike (see the section on the rank). It refused every `glm` before 0.16.0. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. Since 0.17.0 it refuses an `mgcv::gam()` fit with penalised smooth terms: the null distribution of the test is derived for the least-squares residuals of the design of the fit, the basis of its smooths, which the residuals of a penalised fit are not; before, it compared them with that distribution. A `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is tested on that design, as before, and gives the value of the `lm()` of it (decisions 37 and 38 of the design of the model context). Since 0.16.0 the observations of a `gam` are matched to the rows of `data` by the row names of its model frame. Before, they were matched by position, since the residuals of a `gam` carry no names, and `data` with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Harrison, M. J. (1981). A comparison of the bounds, beta-approximate, and exact variants of two tests for heteroscedasticity based on ordinary least squares residuals. *The Economic and Social Review, 12*(4), 235–252.

Harrison, M. J., & McCabe, B. P. M. (1979). A test for heteroscedasticity based on ordinary least squares residuals. *Journal of the American Statistical Association, 74*(366a), 494–499. doi:10.1080/01621459.1979.10482544

Imhof, J. P. (1961). Computing the distribution of quadratic forms in normal variables. *Biometrika, 48*(3-4), 419–426. doi:10.1093/biomet/48.34.419

## See Also

`performGQTest()` and `performSzroeterTest()` for the other tests along an ordering of the observations; `performGlejserTest()` when the errors may be heavy-tailed.

## Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
performHarrisonMcCabeTest(mod, mtcars, order_by = "wt")

# Variance that increases with x
set.seed(7)
x <- runif(80, 1, 5)
d <- data.frame(x = x, y = 1 + 2 * x + rnorm(80, sd = exp(0.3 * x)))
performHarrisonMcCabeTest(lm(y ~ x, data = d), d, order_by = "x")

# Either direction, with a third of the sample in the first part
performHarrisonMcCabeTest(lm(y ~ x, data = d), d,
  order_by = "x",
```

```
  point = 1 / 3, alternative = "two.sided"
)
```

---

`performHartleyFmaxTest` *Perform Hartley's Fmax test*

---

## Description

Compares the maximum and minimum group residual variances using Hartley's maximum F-ratio distribution.

## Usage

```
performHartleyFmaxTest(model, data = NULL, group)
```

## Arguments

| | |
|---|---|
| `model` | A fitted `lm` object. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The residuals and weights are read from the fit itself, and `group` from the rows of `data` that hold the observations of the fit. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`x`, not `log(x)`); a grouping variable the model does not use needs `data`. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `group` | Name of the grouping variable. |

## Details

The statistic is $F_{max} = \max_j s_j^2 / \min_j s_j^2$. Under normality and equal group sizes its null distribution is the maximum F-ratio distribution for $k$ independent mean squares with common degrees of freedom, evaluated with `SuppDists::pmaxFratio()`. If group sizes differ, the function warns and rounds the mean group size, then subtracts one, to obtain an approximate common integer degrees of freedom.

## Value

An object of class `htest` containing the Fmax statistic, number of groups, reference degrees of freedom and p-value.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An `mgcv::gam()` fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context); since 0.16.0 its observations are matched to the rows of `data` by the row names of its model frame. Before, they were matched by position, since the residuals of a `gam` carry no names, and `data` with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Hartley, H. O. (1950). The maximum F-ratio as a short-cut test for heterogeneity of variance. *Biometrika*, 37(3/4), 308–312.

### Examples

```
set.seed(1701)
n <- 20
d <- data.frame(g = factor(rep(letters[1:3], each = n)), x = rnorm(3 * n))
d$y <- 1 + d$x + rnorm(3 * n)
m <- lm(y ~ x, data = d)
performHartleyFmaxTest(m, d, "g")
```

---

## `performHarveyTest` *Perform Harvey test for multiplicative heteroscedasticity*

---

### Description

Harvey's (1976) Lagrange multiplier test, which regresses `log(residuals^2)` on a set of variance regressors.

### Usage

```
performHarveyTest(model, data = NULL, auxiliary = c("regressors", "fitted"),
                  studentize = FALSE)
```

### Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |

| | |
|---|---|
| `data` | optional data frame the model was fitted to. When `NULL`, the default, the model frame of `model` is used. The statistic is computed from the residuals, fitted values and model matrix of `model`, so `data` does not change the result; it is accepted so that every test takes `(model, data, ...)`. Before 0.15.0 the second argument was `auxiliary` and the third `studentize`: a string passed second by position, and a logical value after it, are still read as those two, with a warning that shows the new call; that form stops working in 1.0.0. |
| `auxiliary` | character scalar choosing the variance regressors. `"regressors"` (the default) uses the model's own explanatory variables, the specification in Harvey (1976). `"fitted"` uses the fitted values and their square, which was the behaviour of releases before 0.7.0. |
| `studentize` | logical; if `FALSE` (the default) the chi-square statistic $\mathrm{ESS}/(\pi^2/2)$ is reported, otherwise the auxiliary regression F statistic. |

## Details

Harvey (1976) models the error variance as $\sigma_i^2 = \exp(z_i^\top \gamma)$ and tests $\gamma = 0$ through the auxiliary regression of $\log \hat{e}_i^2$ on $z_i$. Under the null the auxiliary error behaves like a centred $\log \chi_1^2$ variate with variance $\pi^2/2 \approx 4.9348$, so the classical statistic is $\mathrm{ESS}/(\pi^2/2)$, asymptotically chi-square with $q$ degrees of freedom.

Setting `studentize = TRUE` replaces the fixed constant $\pi^2/2$ with the auxiliary residual mean square and refers the overall F statistic to an F distribution. The two forms are asymptotically equivalent under normal errors; the studentized form is more reliable when normality is doubtful, because $\pi^2/2$ is the null variance of $\log \hat{e}^2$ only for Gaussian errors.

Degrees of freedom are taken from the realised rank of the auxiliary design, so a rank-deficient variance model is not credited with degrees of freedom for aliased columns.

## Value

An object of class `htest` containing the test statistic, its degrees of freedom and the p-value.

## Validation

The default statistic reproduces an independent reconstruction of Harvey (1976) to within `1e-8`; see '`tests/testthat/test-pass-a-reference.R`'. Simulated size and power are recorded in '`inst/validation/pass-a-size-power.csv`'.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate `lm` ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without `data`. A fit with `model = FALSE` whose data are gone is

tested on `data` that give its response and fitted values, and refused without them. The test reads nothing from `data`, and since 0.16.0 matches it to the observations of the fit as every test does: data with row names of their own in any order, data that number their rows (a tibble, a data.table) in the order of the fit; other numbered data, rows in front of those of the fit and repeated row names are refused, where it gave the value of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Harvey, A. C. (1976). Estimating regression models with multiplicative heteroscedasticity. *Econometrica*, 44(3), 461–465. doi:10.2307/1913974

Greene, W. H. (2018). *Econometric Analysis* (8th ed.). Pearson. Section 9.5 derives the $\mathrm{ESS}/4.9348$ form of the statistic.

## See Also

`performParkTest`, `performGlejserTest`, `performBPTest`

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performHarveyTest(m)

# Studentized variant, which does not assume normal errors
performHarveyTest(m, studentize = TRUE)

# Variance driven by the conditional mean rather than by the regressors
performHarveyTest(m, auxiliary = "fitted")
```

---

`performHighDimensionalTest` *High-dimensional projection heteroscedasticity test*

---

## Description

Applies principal component projections to the design matrix and evaluates a Breusch-Pagan style statistic on the reduced representation. The procedure is designed for settings where the number of predictors rivals or exceeds the sample size.

## Usage

```
performHighDimensionalTest(
  model,
  data = NULL,
  variance_threshold = 0.9,
  max_components = NULL
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object describing the mean structure whose residual variance is to be assessed. |
| data | The data frame model was fitted to, or NULL, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.16.0; before, such models needed data). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| variance_threshold | Proportion of predictor variance that the selected principal components should explain. Defaults to 0.9. |
| max_components | Optional upper bound on the number of principal components to retain. Defaults to min(10, n - 5). |

## Value

An htest object summarising the chi-squared statistic of the auxiliary regression on principal component scores.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, and data that lack rows or variables of the fit. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## References

Fan, J., & Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B, 70*(5), 849–911.

## Examples

```
set.seed(123)
X <- matrix(rnorm(80 * 20), nrow = 80)
beta <- c(rep(1, 5), rep(0, 15))
y <- X %*% beta + rnorm(80, sd = 0.5 + 0.2 * scale(X[, 1]))
df <- as.data.frame(cbind(y = as.numeric(y), X))
model <- lm(y ~ ., data = df)
performHighDimensionalTest(model, df)
```

## performInfluenceDiagnostics *Identify influential observations*

### Description

Uses Cook's distance to flag influential observations.

### Usage

```
performInfluenceDiagnostics(model, cutoff = NULL)
```

### Arguments

| | |
|---|---|
| model | A fitted lm model. |
| cutoff | Cook's distance threshold; defaults to 4/(n - p). |

### Value

A list with Cook's distances and indices of influential points.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performInfluenceDiagnostics(m)
```

## performKoenkerTest *Perform Koenker studentized Breusch-Pagan test*

### Description

Implementation of the Koenker version of the Breusch-Pagan test.

### Usage

```
performKoenkerTest(model, data = NULL)
```

## Arguments

| | |
|---|---|
| model | an object of class lm. |
| data | The data frame model was fitted to, or NULL, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.16.0; before, such models needed data). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |

## Details

Squared residuals are regressed on the regressors but the statistic $nR^2$ uses a studentized form that is robust to non-normality.

The test is computed in memory whatever the size of the data. Up to 0.15.0 it switched to performKoenkerTestStreaming() on data above 100 MB; call that function to stream.

## Value

An object of class htest containing the test statistic, p-value and degrees of freedom.

## Fixed effects for small groups

Koenker's test, like White's and the Breusch-Pagan tests, is not valid with fixed effects for small groups: a factor with many levels and few observations of each, such as subject in lm(y ~ x + subject) with each subject observed a few times. On homoscedastic data it rejects too often, and the more often the more levels the factor has and the fewer observations each level has. The excess comes from the dummies of the factor in the auxiliary regression. The residuals of a level are taken from its own mean, so under normal errors the squared residuals of a level of $T$ observations vary between levels $T/(T-1)$ times as much as the auxiliary regression assumes, and with $G$ levels the statistic has mean its degrees of freedom plus $(G-1)/(T-1)$, as it had in the study below, within 2.3% for $T$ of 3 or more. Regressed on x alone, without the dummies, the squared residuals were rejected in 4.6% to 5.5% of the draws (inst/validation/fixed-effects-dummies.R). The excess grows with $G$ at a fixed $T$, while the spread of the chi-squared distribution grows as $\sqrt{2G}$, so with a fixed number of observations per level the size goes to 1 as the number of levels grows.

Since 0.16.0 the test warns when a term of the model that holds a factor or a character variable has more levels than observations per level: $L > n/L$, that is $L^2 > n$, with $L$ the number of distinct combinations of the factors of the term among the $n$ observations of the fit. The term is a factor, such as subject; an interaction of factors, counted by its cells, such as a:b of a crossed a * b or school:class of a nested school / class; or a slope for each level, x:subject. A variable left out of the formula, as in y ~ . - id, does not count. The warning names the term and gives $L$, $n$ and the mean number of observations per level; the statistic, the degrees of freedom and the p-value are those the test computes without it. It has class heteroTests_fixed_effects_warning, which suppressWarnings(expr, classes = "heteroTests_fixed_effects_warning") muffles, and no

other warning. It is given on every call, also when `cachedTest()` or `runHeteroTests()` answers from its cache, and `runHeteroTests()` and `runDiagnostics()` give it once for each test that gives it. `performKoenkerTestStreaming()`, `performBPTestRobust()` with `studentize = TRUE`, the deprecated `performStudentizedBPTest()` and the registered `"koenker"` and `"student_bp"` give it too. `performWildBootstrapTest()`, which bootstraps the same statistic, held its level on these designs and does not give it.

On a balanced design of $G$ subjects of $T$ observations the rule is $G \leq T$. In the study it kept the three tests within 2.5% to 7.5% at the 5% level where it holds, and put at least one of them outside where it does not, in 68 of the 69 designs where they computed; the exception is the Breusch-Pagan test on 20 subjects of 20, at 8.0%. Unbalanced groups, a second regressor and a regressor correlated with the subject effects followed the same pattern, with the mean number of observations per level, $n/G$, in place of $T$. The rule is a warning, not a correction, and many panels fail it: on 100 firms observed over 10 years the three tests reject in 18% to 21% of the draws. Rejection at the 5% level, in percent, in 2,000 homoscedastic draws of `lm(y ~ x + subject)`, with `x`, the subject effects and the errors standard normal:

| Subjects | Observations of each | White | Koenker | Breusch-Pagan | Warns |
|---|---|---|---|---|---|
| 10 | 10 | 6.4 | 6.2 | 7.3 | no |
| 20 | 20 | 6.0 | 6.4 | 8.0 | no |
| 5 | 4 | 1.0 | 5.4 | 7.0 | yes |
| 30 | 3 | 50.7 | 57.3 | 39.6 | yes |
| 30 | 4 | 22.2 | 23.9 | 28.7 | yes |
| 100 | 8 | 23.8 | 23.8 | 25.2 | yes |
| 100 | 10 | 18.1 | 18.4 | 21.3 | yes |
| 200 | 4 | 97.8 | 97.6 | 82.5 | yes |

A slope for each level counts too, but there the error goes the other way: with a slope for each subject, `lm(y ~ x + x:subject)`, on 30 subjects of 3 and of 4 and on 100 of 8, the three tests rejected too rarely, White's in 0.7% to 2.2% of the draws, Koenker's in 0.1% to 0.8% and the Breusch-Pagan test in 0.4% to 2.8% (`inst/validation/fixed-effects-variants-size.R`). The warning therefore says that the p-value is unreliable: in the simulations usually far too small, and too large with a slope per level.

The study is `inst/validation/fixed-effects-size.R`, with its results in `fixed-effects-size.csv` (`system.file("validation", package = "heteroTests")`). Juhl and Sosa-Escudero (2014) derive a test of heteroskedasticity for this setting, after fixed effects estimation of a panel of many units observed a fixed number of times; it needs to be told which factor is the unit, and the package does not implement it.

### Weighted fits

`lm(..., weights = w)` states that the error variance is $\sigma^2/w_i$. On such a fit the test is computed from the Pearson residuals $\sqrt{w_i}\, e_i$, the residuals of the equivalent unweighted regression, which have constant variance when the weights are right. The null hypothesis is then that the weights are adequate; the variance regressors stay on their original scale. Releases before 0.12.0 used the raw residuals, whose variance differs across observations by assumption, so a correctly weighted fit was rejected as often as an unweighted one. A fit with a zero weight is refused.

The reference distribution takes the weights as known. When they were estimated from the same data, as by `fitWLS()`, it is only approximate, and not more accurate in larger samples. With a correctly specified variance function and normal errors, the Koenker test rejected 8% of the time at the 5% level with 150 observations and 11% with 600, the White test 6% and 8%, and the Harvey test never, because it repeats the regression the weights were estimated from (`inst/validation/weighted-fits-size.csv`). `performVarianceFormTest()` accounts for the estimation, held its level in the same designs, and is the test to use on a `fitWLS()` fit.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Koenker, R. (1981). A note on studentizing a test for heteroscedasticity. *Journal of Econometrics*, 17(1), 107–112.

Juhl, T., & Sosa-Escudero, W. (2014). Testing for heteroskedasticity in fixed effects models. *Journal of Econometrics*, 178(Part 3), 484–494. doi:10.1016/j.jeconom.2013.07.005

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performKoenkerTest(m, mtcars)
```

---

# performKoenkerTestStreaming

*Streaming Koenker studentized Breusch-Pagan test*

---

## Description

Evaluates Koenker's studentized variant of the Breusch-Pagan test (the $nR^2$ statistic from regressing the squared residuals on the regressors) chunk by chunk so that large datasets can be processed without allocating the full auxiliary regression matrix.

## Usage

```
performKoenkerTestStreaming(
  model,
  data = NULL,
  chunk_size = 10000,
  progress = interactive()
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object supplying residuals and fitted values for the diagnostic. |
| data | The data frame model was fitted to, or NULL, the default, read as performWhiteTestStreaming() reads it: the response, design, weights and offset come from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data (since 0.16.0), and data that are given are matched to the observations of the fit. |
| chunk_size | Positive integer giving the number of observations processed per streaming chunk. |
| progress | Logical flag indicating whether a textual progress bar should be shown while processing the chunks. Defaults to interactive(). |

## Details

Each chunk of the regressors, centred at their means, is reduced by a QR decomposition stacked on the triangular factor of the chunks before it, as in performWhiteTestStreaming(), since 0.17.0. Up to 0.16.0 the cross-products of the regressors were accumulated: those of a regressor 1,000 standard deviations from zero looked collinear with the intercept, and the test stopped ("Koenker auxiliary regression became rank deficient"); and squared residuals of about $10^{-10}$ were taken for constant ("requires variation in squared residuals"). It gives the value of performKoenkerTest() on both. Sparse cross-products from **Matrix** are no longer used.

Since 0.16.0 the chunks are rows of the design of the fit, as in performWhiteTestStreaming(), whose details say what changed.

## Value

A htest object equivalent to performKoenkerTest with metadata recording the streaming configuration.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, data that lack rows or variables of the fit, and, since 0.16.0, a multivariate lm, a fit changed after it was fitted and a fit with model = FALSE whose data are gone, unless data gives its response and fitted values. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## See Also

performKoenkerTest for the standard implementation.

## Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
performKoenkerTestStreaming(mod, mtcars, chunk_size = 16, progress = FALSE)
```

`performLeveneTest` *Perform Levene's test for equality of variances*

## Description

Tests equality of variances across groups using residuals from a linear model.

## Usage

```
performLeveneTest(model, data = NULL, group)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The residuals and weights are read from the fit itself, and `group` from the rows of `data` that hold the observations of the fit; the observations whose group is missing are left out, with a warning. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`x`, not `log(x)`); a grouping variable the model does not use needs `data`. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `group` | name of the grouping variable. |

## Details

Absolute deviations from group means are analyzed via a one-way ANOVA. The resulting $F$ statistic has $k - 1$ and $N - k$ degrees of freedom under the null of equal variances.

## Value

An object of class `htest` containing the F statistic, p-value and degrees of freedom.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An `mgcv::gam()` fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37

of the design of the model context); since 0.16.0 its observations are matched to the rows of `data` by the row names of its model frame. Before, they were matched by position, since the residuals of a `gam` carry no names, and `data` with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

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

Brown, M. B., & Forsythe, A. B. (1974). Robust tests for the equality of variances. *Journal of the American Statistical Association*, 69(346), 364–367. doi:10.1080/01621459.1974.10482955

## Examples

```
data(mtcars)
mtcars$cyl <- factor(mtcars$cyl)
m <- lm(mpg ~ wt, data = mtcars)
performLeveneTest(m, mtcars, "cyl")
```

---

# `performMcLeodLiTest` *Perform McLeod-Li test*

---

## Description

Applies the Ljung-Box test to squared residuals.

## Usage

```
performMcLeodLiTest(model, data = NULL, lags = 10)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | optional data frame the model was fitted to. When `NULL`, the default, the model frame of `model` is used. The statistic is computed from the residuals of `model`, so `data` does not change the result; it is accepted so that every test takes `(model, data, ...)`. Before 0.15.0 the second argument was `lags`: a number passed there by position is still read as `lags`, with a warning that shows the new call; that form stops working in 1.0.0. |
| `lags` | number of lags for the Ljung-Box test. |

## Details

Serial correlation in $e_t^2$ indicates conditional heteroscedasticity. The Ljung-Box statistic is compared to a chi-square distribution with the chosen number of lags.

## Value

An object of class `htest` containing the test statistic, p-value and degrees of freedom.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate `lm` ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without `data`. A fit with `model = FALSE` whose data are gone is tested on `data` that give its response and fitted values, and refused without them. The test reads nothing from `data`, and since 0.16.0 matches it to the observations of the fit as every test does: data with row names of their own in any order, data that number their rows (a tibble, a data.table) in the order of the fit; other numbered data, rows in front of those of the fit and repeated row names are refused, where it gave the value of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

McLeod, A. I., & Li, W. K. (1983). Diagnostic checking ARMA time series models using squared-residual autocorrelations. *Journal of Time Series Analysis*, 4(4), 269–273.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performMcLeodLiTest(m, lags = 10)
```

---

# `performNCVTest` *Perform the non-constant variance (NCV) score test (deprecated)*

---

## Description

The Cook-Weisberg score test for non-constant error variance, the diagnostic returned by `car::ncvTest()`.

## Usage

```
performNCVTest(model, data = NULL, var_formula = NULL)
```

## Arguments

| | |
|---|---|
| model | an object of class lm. |
| data | optional data frame the model was fitted to, in which var_formula is evaluated, as in performCookWeisbergTest. When NULL, the default, the model frame of model is used. |
| var_formula | optional one-sided formula giving the variance model, evaluated in data (for example ~ x1 + x2). When NULL the fitted values are used. Before 0.15.0 var_formula was the second argument: a formula passed there by position is still read as var_formula, with a warning; that form stops working in 1.0.0. |

## Details

Let $\hat{e}_i$ denote the OLS residuals and $\hat{\sigma}^2 = \sum_i \hat{e}_i^2/n$ the maximum-likelihood variance estimate. The test regresses the scaled squared residuals $u_i = \hat{e}_i^2/\hat{\sigma}^2$ on the variance regressors $Z$ and computes half the explained sum of squares of that auxiliary regression. Under homoscedasticity and normal errors the statistic is asymptotically chi-square with degrees of freedom equal to the number of variance regressors.

With var_formula = NULL the fitted values are the sole variance regressor, matching the default of car::ncvTest() and Stata's estat hettest.

Because the divisor 2 is the null variance of $\hat{e}^2/\sigma^2$ under normality, the test is sensitive to non-normal errors; performKoenkerTest provides the studentized variant, which is robust to non-normal kurtosis.

## Value

An object of class htest containing the chi-square score statistic, its degrees of freedom and the p-value.

## Deprecated

performNCVTest() is deprecated since 0.14.0: use performCookWeisbergTest, which gained var_formula and is the one entry point for the Cook-Weisberg score test. With a variance formula, performNCVTest(model, var_formula) returns what performCookWeisbergTest(model, var_formula) returns; without one the two return the same statistic, degrees of freedom and p-value, and performCookWeisbergTest() labels the result as the Cook-Weisberg test. The old name still works, returns what it returned and warns.

## Weighted fits

On a fit with weights the test is computed from the Pearson residuals $\sqrt{w_i}\, e_i$, so it tests whether the weights are adequate; see the section on weighted fits in performKoenkerTest(), which also covers weights estimated by fitWLS().

## Validation

Reproduces car::ncvTest() to within 1e-8 for both the default and the var_formula forms; see 'tests/testthat/test-pass-a-reference.R'. Releases before 0.7.0 regressed the *absolute*

residuals on the fitted values and reported a t statistic, which is a Glejser-type test rather than the Cook-Weisberg score test.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate `lm` ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without `data`. A fit with `model = FALSE` whose data are gone is tested on `data` that give its response and fitted values, and refused without them. `data` is matched to the observations of the fit as by `performCookWeisbergTest`, with or without `var_formula`. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Cook, R. D., & Weisberg, S. (1983). Diagnostics for heteroscedasticity in regression. *Biometrika*, 70(1), 1–10. doi:10.1093/biomet/70.1.1

Fox, J., & Weisberg, S. (2019). *An R Companion to Applied Regression* (3rd ed.). SAGE.

### See Also

`performCookWeisbergTest`, `performKoenkerTest`, `performBPTest`

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
# Deprecated: use performCookWeisbergTest(m) and
# performCookWeisbergTest(m, var_formula = ~wt)
suppressWarnings(performNCVTest(m, var_formula = ~wt))
```

---

## `performOBrienTest` *Perform O'Brien test for equality of variances*

---

### Description

Implements O'Brien's observation-level transformed-residual test for equality of group variances.

### Usage

```
performOBrienTest(model, data = NULL, group)
```

## Arguments

| | |
|---|---|
| model | A fitted lm object. |
| data | The data frame model was fitted to, or NULL, the default. The residuals and weights are read from the fit itself, and group from the rows of data that hold the observations of the fit; the observations whose group is missing are left out, with a warning. Without data it is read from the model frame of model, which holds it only when it is a variable of the model under its own name (x, not log(x)); a grouping variable the model does not use needs data. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| group | Grouping variable name. |

## Details

For each residual within a group, O'Brien's transformation combines its squared deviation from the group mean with the group's sample variance and sample size. A one-way ANOVA of those transformed observations yields an approximate $F_{k-1,N-k}$ test. Every group must contain at least three observations.

## Value

An object of class htest containing the F statistic, numerator and denominator degrees of freedom, and p-value.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, and data that lack rows or variables of the fit. It refuses a multivariate lm ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An mgcv::gam() fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context); since 0.16.0 its observations are matched to the rows of data by the row names of its model frame. Before, they were matched by position, since the residuals of a gam carry no names, and data with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## References

O'Brien, R. G. (1981). A simple test for variance effects in experimental designs. *Psychological Bulletin*, 89(3), 570–574.

## Examples

```
set.seed(1702)
n <- 25
d <- data.frame(g = factor(rep(letters[1:3], each = n)), x = rnorm(3 * n))
d$y <- 2 + d$x + rnorm(3 * n, sd = rep(c(1, 1, 1.8), each = n))
m <- lm(y ~ x, data = d)
performOBrienTest(m, d, "g")
```

---

`performParkTest` *Perform Park test for heteroscedasticity*

---

## Description

Regresses the log of squared residuals on the log of a suspected variable.

## Usage

```
performParkTest(model, data = NULL, variable)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The residuals and weights are read from the fit itself, and `variable` from the rows of `data` that hold the observations of the fit. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`x`, not `log(x)`); a model that transforms the variable needs `data`, as does a variable the model does not use. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `variable` | name of the suspected explanatory variable. |

## Details

The slope coefficient from $\log(e^2)$ regressed on $\log x$ is tested for zero using a t statistic. A significant coefficient indicates scale depending on $x$.

## Value

An object of class `htest` containing the t statistic, p-value and degrees of freedom.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An `mgcv::gam()` fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context); since 0.16.0 its observations are matched to the rows of `data` by the row names of its model frame. Before, they were matched by position, since the residuals of a `gam` carry no names, and `data` with the rows of the fit in another order gave a wrong value. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Park, R. E. (1966). Estimation with heteroscedastic error terms. *Econometrica*, 34(4), 888. doi:10.2307/1910108

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performParkTest(m, mtcars, "wt")
```

---

## `performPesaranTest` *Perform Pesaran CD test*

---

### Description

Checks for cross-sectional dependence in panel residuals.

### Usage

```
performPesaranTest(model, data = NULL, id, time)
```

### Arguments

| | |
|---|---|
| `model` | an object of class `lm`, the pooled `lm()` of the panel. |
| `data` | The data frame `model` was fitted to, holding the panel identifiers named by `id` and `time`, or `NULL`, the default. The residuals and weights are read from the fit itself, and `id` and `time` from the rows of `data` that hold the observations of the fit. Without `data` they are read from the model frame of `model`, which holds them only when they are variables of the model under their own names (`id`, not `factor(id)`); identifiers the model does not use need `data`. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a |

data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`.

`id` individual identifier column.

`time` time column. Each pair of `id` and `time` must hold one observation of the fit: since 0.16.0 a pair that holds more is refused. The units need not be observed at the same periods (Details).

## Details

The statistic averages pairwise correlation coefficients of the residuals and is standardized to follow a normal distribution under the null of independence. On a balanced panel of $N$ units observed at the same $T$ periods it is

$$\mathrm{CD} = \sqrt{\frac{2T}{N(N-1)}} \sum_{i<j} \hat{\rho}_{ij},$$

where $\hat{\rho}_{ij}$ is the correlation of the residuals of units $i$ and $j$. On an unbalanced panel, whose units are not all observed at the same periods, it is computed, since 0.16.0, from the periods each pair of units shares (Pesaran 2004, Section 9, equation 31), as `plm::pcdtest(test = "cd")` computes it:

$$\mathrm{CD} = \sqrt{\frac{1}{P}} \sum_{i<j} \sqrt{T_{ij}}\, \hat{\rho}_{ij},$$

where $T_{ij}$ is the number of periods at which units $i$ and $j$ are both observed, $\hat{\rho}_{ij}$ the correlation of their residuals over those periods, each centred at its mean over them, and $P$ the number of pairs that share at least two periods, the pairs of the sum: $N(N-1)/2$ when every pair does. On a balanced panel, where $T_{ij} = T$, the two are the same statistic. A pair that shares fewer than two periods has no correlation and is left out, with a warning, as `plm::pcdtest()` leaves it out, and a panel in which no pair shares two periods is refused. A correlation that is not defined, the residuals of a unit constant over the periods of a pair, adds nothing to the sum and the pair stays in the count, as on a balanced panel, with a warning; `plm::pcdtest()` returns `NA` there, so the statistic is that of `plm::pcdtest()` except where a correlation is undefined. Pesaran (2004) states its standard normal limit under the null of independence for $T_i > k+1$ and $T_{ij} > 3$, with $N$ large, $T_i$ being the number of periods of unit $i$ and $k$ the number of regressors of its own regression.

## Value

An object of class `htest`.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. The model is a pooled `lm` of the panel, with the units and the periods as columns of `data`; a `plm::plm()` fit is not an `lm` and

is refused. Since 0.16.0 a pair of `id` and `time` that holds two observations of the fit is refused. An unbalanced panel, as that of a fit that dropped some rows of some units, is tested since 0.16.0, from the periods each pair of units shares (Details), and gives the value of `plm::pcdtest(test = "cd")` on the residuals of the fit, except where a correlation is undefined (Details): before, the test stopped on it, or, when the number of periods of a unit divided that of the panel, returned a wrong value computed on its residuals recycled over the periods it lacks. A fit with `na.action = na.exclude` is tested since 0.17.0 as the same fit with `na.omit`, on the observations it kept, its panel balanced or not: before, it was read by position, its residuals padded with `NA` at the rows it dropped, against the rows of `data`, which had to be in the order of the fit, and an unbalanced panel of those rows was refused. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Pesaran, M. H. (2004). General diagnostic tests for cross section dependence in panels. *CESifo Working Paper* 1229 (also *Cambridge Working Papers in Economics* 0435). Section 9, "CD tests in the case of unbalanced panels", equation 31.

Pesaran, M. H. (2015). Testing weak cross-sectional dependence in large panels. *Econometric Reviews* 34(6-10), 1089–1117. doi:10.1080/07474938.2014.956623.

Croissant, Y. and Millo, G. (2008). Panel data econometrics in R: the **plm** package. *Journal of Statistical Software* 27(2), 1–43. doi:10.18637/jss.v027.i02. The statistic of an unbalanced panel is that of `plm::pcdtest(test = "cd")` (plm 2.6-6), whose help page says its tests are "suitable also for unbalanced panels".

## Examples

```
df <- data.frame(id = rep(1:3, each = 5), time = rep(1:5, 3), x = runif(15), y = rnorm(15))
m <- lm(y ~ x, data = df)
performPesaranTest(m, df, "id", "time")
```

---

`performQuantileRegressionTest`

*Quantile regression heteroscedasticity test*

---

## Description

Tests equality of regression slopes across two or more conditional quantiles using the joint Wald-type test implemented by `quantreg::anova.rqs()`. The procedure accounts for dependence among quantile-specific estimates from the same sample rather than treating their covariance matrices as independent.

## Usage

```
performQuantileRegressionTest(
  model,
  data = NULL,
  taus = c(0.25, 0.75),
```

```
  se_type = c("nid", "ker"),
  iid = TRUE
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object describing the mean structure whose conditional quantile slopes are to be compared. |
| data | The data frame model was fitted to, or NULL, the default. The response, design and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.16.0; before, such models needed data). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| taus | Numeric vector containing at least two distinct quantiles strictly between zero and one. |
| se_type | Standard-error method used by quantreg::anova.rqs(); supported values are "nid" and "ker". |
| iid | Logical indicating whether identical conditional densities are assumed when computing the joint test. |

## Details

Under a pure location-shift model with homoskedastic errors, regression slopes are equal across quantiles. Rejection therefore provides evidence against that location-shift/homoskedastic specification. The result should not be interpreted as a universal test for every possible form of heteroscedasticity.

Weighted fits are refused, although the resampling tests accept them. On a fit with weights $w_i$ the null hypothesis is that the Pearson residuals $\sqrt{w_i}\, e_i$ have constant variance, which is a statement about the regression of $\sqrt{w_i}\, y_i$ on $\sqrt{w_i}\, x_i$. The quantile regressions this test fits are of $y$ on $x$, where the errors have variance $\sigma^2/w_i$ by assumption, so their slopes differ across quantiles even when the weights are right. Fitting them on the weighted scale does not by itself repair that: there the conditional quantiles are the regression plus a constant, and the constant is not a column of the weighted regressors. Adding a column of ones would restore the location-shift form, but that version of the test needs its own derivation and its own size study, and has neither.

## Value

An htest object containing the F-like joint statistic, numerator and denominator degrees of freedom, p-value, fitted quantiles, and quantile-specific slope estimates.

### Supported models

It accepts an unweighted `lm` and an unweighted `glm` with the Gaussian family and the identity link and, since 0.16.0, a rank-deficient one: `quantreg::rq()` refuses a singular design, so the quantile regressions are fitted to the columns that were not aliased, which gives the value of the same model fitted without them. Since 0.16.0 it also accepts a model with an offset: the quantile regressions are fitted to the response of the fit minus its offset, on the design of the fit, which is the model with the offset, since the offset shifts every conditional quantile as it shifts the mean; before, they were refitted from the formula, which left the offset out. Since 0.16.0 it also accepts a model with a single regressor, such as `lm(y ~ x)`; before, it stopped with "subscript out of bounds" on one, and `runHeteroTests()` returned its fallback, the rank permutation test, under the name `quantile_regression`. It refuses a weighted fit, a `fitWLS()` fit among them, as it refits the model without the weights, a `glm` of another family or link, a `survey::svyglm()` fit, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), and, since 0.16.0, an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, the basis of its smooths, that the test fits again. Since 0.17.0 a `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is tested on that design and gives the value of the `lm()` of it; before, it was refused as a penalised one. A `gam` without smooth terms is tested as the Gaussian `glm` it is (decisions 37 and 38 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

Koenker, R., & Bassett, G. (1982). Robust tests for heteroscedasticity based on regression quantiles. *Econometrica, 50*(1), 43–61.

Koenker, R. (2005). *Quantile Regression*. Cambridge University Press.

### Examples

```
if (requireNamespace("quantreg", quietly = TRUE)) {
  # The test needs at least 40 observations, so mtcars (32) is too small.
  model <- lm(stations ~ mag + depth, data = quakes)
  performQuantileRegressionTest(model, quakes)
}
```

---

## performRankPermutationTest

*Rank-based permutation heteroscedasticity test*

---

### Description

Performs a non-parametric permutation test based on the rank correlation between absolute residuals and a chosen ordering variable. Significant correlations imply systematic changes in residual spread.

## Usage

```
performRankPermutationTest(
  model,
  data = NULL,
  order_by = NULL,
  B = 999,
  progress = interactive()
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object describing the mean structure whose residual variance is to be assessed. |
| data | The data frame model was fitted to, or NULL, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.16.0; before, such models needed data), unless order_by names a variable the model frame does not hold, which is then read from data. Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation and, when order_by is read from data, no two rows of the fit tie on every column of its model frame. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| order_by | Optional name of a predictor variable used to rank the observations. Defaults to the first non-intercept term in the model matrix. |
| B | Number of permutation replications used to approximate the null distribution. Defaults to 999, the default of every resampling test in the package. |
| progress | Logical toggle for progress reporting during permutations. |

## Value

An htest object containing the observed Spearman correlation and a permutation-based p-value.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, and data that lack rows or variables of the fit. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## References

Hollander, M., Wolfe, D. A., & Chicken, E. (2013). *Nonparametric Statistical Methods* (3rd ed.). Wiley.

## Examples

```
data(mtcars)
model <- lm(mpg ~ wt + hp, data = mtcars)
set.seed(42)
performRankPermutationTest(model, mtcars, B = 199, progress = FALSE)
```

---

# performRESETTest *Ramsey's RESET test for nonlinearity*

---

## Description

Adds powers of the fitted values and performs an F-test to check for neglected nonlinearity.

## Usage

```
performRESETTest(model, data = NULL, power = 2:3)
```

## Arguments

| | |
|---|---|
| model | A fitted lm model. |
| data | optional data frame the model was fitted to, or NULL, the default. The statistic is computed from the response, fitted values, design, weights and offset of the fit itself, so data does not change the result; it is accepted so that every test takes (model, data, ...); since 0.16.0 its rows are matched to the observations of the fit as by every test, data with row names of their own in any order and data that number their rows in the order of the fit, and other data are refused, where it needed only the variables of the model and the row names of the fit. Before 0.15.0 the second argument was power: a number passed there by position is still read as power in 0.15.x, with a warning that shows the new call, and stops working in 1.0.0. |
| power | Numeric vector of powers to include. |

## Details

On a fit with weights the augmented model is fitted by weighted least squares and both residual sums of squares are weighted. Releases before 0.12.0 refitted the augmented model without the weights, so the two models were not nested.

On a fit with an offset the fitted values are the design times the coefficients plus the offset, and the test is the nested F test of the model with its offset against the model augmented with the powers of those fitted values, fitted with the same offset and weights (since 0.16.0, decided 2026-10-18). It is

not RESET of the model fitted to the response minus the offset, which augments that model with the powers of the design times the coefficients alone. Before 0.16.0 the augmented model was fitted without the offset, so the two models were not nested.

### Value

An object of class `htest`.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, weighted or not, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. A fit with a zero weight is accepted too: the test compares two weighted least-squares fits, which both leave the observation out, and gives the value of the fit to the observations with positive weight (kept so, decided on 2026-10-17). It accepts a model with an offset, given as an argument or as an `offset()` term, weighted or not, and tests the model with its offset (since 0.16.0; see *Details*). It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a coefficient that is not finite for another reason, `data` that lack rows or variables of the fit, and, since 0.16.0, a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit: it read that frame and returned a value. A fit with `model = FALSE` whose data are gone is tested on `data` that give its response and fitted values (since 0.16.0; it refused such a fit until the tests computed from the model alone were built on the model context), and refused without them. Since then it also refuses numbered data in another order than the fit, rows in front of those of the fit and repeated row names, as every test does, where it gave the value of the fit. It refuses a multivariate `lm` ("needs a model with one response"), which it tested without `data` before 0.16.0, and, since 0.16.0, an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, the basis of its smooths, that the test fits again. Since 0.17.0 a `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is tested on that design and gives the value of the `lm()` of it; before, it was refused as a penalised one. A `gam` without smooth terms is tested as the Gaussian `glm` it is (decisions 37 and 38 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performRESETTest(m)
```

---

## performScatterDiagnostic

*Scatter-plot diagnostics for heteroscedasticity*

---

### Description

Computes correlations between absolute residuals and selected variables.

## Usage

```
performScatterDiagnostic(model, data, vars)
```

## Arguments

| | |
|---|---|
| model | an object of class lm. |
| data | The data frame model was fitted to, holding the variables named in vars. The residuals and weights are read from the fit itself, and vars from the rows of data that hold the observations of the fit. Since 0.16.0 they are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| vars | character vector of variable names. |

## Details

High correlations suggest increasing spread with the explanatory variables and motivate variance-stabilizing transformations.

## Value

A named numeric vector of correlations.

## Supported models

It accepts the models the tests accept: an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and a rank-deficient fit. It refuses a glm of another family or link (since 0.16.0), a survey::svyglm() fit, a fit with a zero weight, and data that lack rows or (since 0.16.0) variables of the fit. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## References

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368), 829–836.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performScatterDiagnostic(m, mtcars, c("wt", "qsec"))
```

`performSpatialHeteroTest` *Spatial heteroscedasticity test*

## Description

Evaluates spatial clustering in squared residuals using Moran's I with a permutation reference distribution. Significant positive autocorrelation in the squared residuals is evidence of spatially varying variance.

## Usage

```
performSpatialHeteroTest(
  model,
  data = NULL,
  listw,
  permutations = 999,
  zero.policy = NULL
)
```

## Arguments

| | |
|---|---|
| `model` | A fitted stats::lm object describing the mean structure whose residual variance is to be assessed. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The residuals and weights are read from the fit itself, so a model that transforms its variables, has an `offset()` term, or takes its `weights` or `offset` from a column of the data needs no `data`, and the model frame of `model` is accepted as `data` (since 0.16.0; before, such models needed `data`). The test reads no column of `data`, and `listw` is matched to the observations of the fit, not to the rows of `data`. Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `listw` | Spatial weights in spdep `listw` format or a numeric matrix coercible via `spdep::mat2listw()`. They list the observations of the fit, one per residual, in the order of the rows `model` was fitted to after `subset` and after `na.omit` dropped incomplete rows, whatever the order of `data`, as before 0.16.0. |
| `permutations` | Number of Monte Carlo permutations used to compute the reference distribution. Defaults to 999, the default of every resampling test in the package since 0.15.0; it was 499 before. |
| `zero.policy` | Logical flag forwarded to the spatial diagnostic to permit islands with no neighbours. |

## Value

An `htest` result with Moran's I statistic applied to squared residuals.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. It refuses a multivariate `lm` ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. An `mgcv::gam()` fit is tested as before 0.16.0, with penalised smooth terms too: the test reads its residuals and does not fit it again, so it does not take them for least-squares residuals (decision 37 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Anselin, L. (1988). *Spatial Econometrics: Methods and Models*. Kluwer.

Bivand, R. S., Pebesma, E., & Gómez-Rubio, V. (2013). *Applied Spatial Data Analysis with R* (2nd ed.). Springer.

## Examples

```
if (requireNamespace("spdep", quietly = TRUE)) {
  data(mtcars)
  coords <- cbind(runif(nrow(mtcars)), runif(nrow(mtcars)))
  nb <- spdep::knn2nb(spdep::knearneigh(coords, k = 4))
  lw <- spdep::nb2listw(nb)
  model <- lm(mpg ~ wt + hp, data = mtcars)
  performSpatialHeteroTest(model, mtcars, listw = lw, permutations = 199)
}
```

---

# performSpearmanTest *Perform Spearman rank correlation test*

---

## Description

Computes Spearman's rho between absolute residuals and fitted values.

## Usage

```
performSpearmanTest(model, data = NULL)
```

## Arguments

| | |
|---|---|
| model | an object of class lm. |
| data | optional data frame the model was fitted to. When NULL, the default, the model frame of model is used. The statistic is computed from the residuals and fitted values of model, so data does not change the result; it is accepted so that every test takes (model, data, ...). |

## Details

The squared rank correlation coefficient times the sample size minus 1 approximates a chi-square distribution with one degree of freedom.

## Value

An object of class htest containing the test statistic, p-value, degrees of freedom and rho.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, and data that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate lm ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with model = FALSE whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without data. A fit with model = FALSE whose data are gone is tested on data that give its response and fitted values, and refused without them. The test reads nothing from data, and since 0.16.0 matches it to the observations of the fit as every test does: data with row names of their own in any order, data that number their rows (a tibble, a data.table) in the order of the fit; other numbered data, rows in front of those of the fit and repeated row names are refused, where it gave the value of the fit. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## References

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15(1), 72–101.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performSpearmanTest(m)
```

`performSpreadLevelTest` *Perform Spread-Level test*

## Description

Tests the slope of `log(|residuals|)` on `log(fitted)`.

## Usage

```
performSpreadLevelTest(model, data = NULL)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | optional data frame the model was fitted to. When `NULL`, the default, the model frame of `model` is used. The statistic is computed from the residuals and fitted values of `model`, so `data` does not change the result; it is accepted so that every test takes `(model, data, ...)`. |

## Details

A slope near 0 indicates constant variance. A significant slope suggests a power transformation may stabilize the spread.

## Value

An object of class `htest`.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate `lm` ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after the fit, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit, with or without `data`. A fit with `model = FALSE` whose data are gone is tested on `data` that give its response and fitted values, and refused without them. The test reads nothing from `data`, and since 0.16.0 matches it to the observations of the fit as every test does: data with row names of their own in any order, data that number their rows (a tibble, a data.table) in the order of the fit; other numbered data, rows in front of those of the fit and repeated row names are refused, where it gave the value of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368), 829–836.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performSpreadLevelTest(m)
```

---

`performStudentizedBPTest`

*Studentized Breusch–Pagan test (deprecated)*

---

## Description

`performStudentizedBPTest()` is deprecated since 0.14.0: use `performKoenkerTest()`, which computes the same statistic. It still works and warns, and returns what it returned before except on a fit returned by `fitWLS()`. The model frame of that fit holds the weights of the model `fitWLS()` refitted (none, for an unweighted model), not the estimated weights, and before 0.15.0 `performStudentizedBPTest()` read them from there, so it computed the statistic of the unweighted fit. Since 0.15.0 it reads the weights of the fit, as `performKoenkerTest()` does, and the two agree. On a model with an offset they agree since 0.16.0: before, the refit of the design left the offset out, and `performStudentizedBPTest()` returned the statistic of another model.

## Usage

```
performStudentizedBPTest(model, data = NULL)
```

## Arguments

| | |
|---|---|
| `model` | A fitted stats::lm object providing the residuals and design matrix for the auxiliary regression. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an `offset()` term, or takes its `weights` or `offset` from a column of the data needs no `data`, and the model frame of `model` is accepted as `data` (since 0.16.0; before, such models needed `data`). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |

## Details

Computes Koenker's (1981) studentized form of the Breusch–Pagan Lagrange multiplier statistic, $nR^2$ from the regression of the squared residuals on the regressors. The classical Breusch–Pagan statistic assumes normal errors and this one does not: in the simulations of `inst/validation/` it holds its level under $t_5$ errors, where the classical statistic rejects far too often.

Following Koenker (1981) and the implementation in lmtest::bptest(), the procedure fits an auxiliary regression of $e_i^2 - \hat{\sigma}^2$ on the regressors from the original model (including the intercept), where $e_i$ denotes the residuals and $\hat{\sigma}^2 = \sum e_i^2/n$ the mean of their squares. With $\hat{g}_i$ the fitted values of that regression, the statistic $T = n \sum \hat{g}_i^2 / \sum (e_i^2 - \hat{\sigma}^2)^2$ is its $nR^2$. Under homoskedasticity it is asymptotically chi-squared with degrees of freedom equal to the number of regressors beyond the intercept, whether or not the errors are normal. On a weighted fit $e_i$ is the Pearson residual $\sqrt{w_i}$ times the raw residual and the auxiliary regression is unweighted; see the section on weighted fits in `performKoenkerTest()`. This departs from `lmtest::bptest()`, which keeps the raw residuals and weights the auxiliary regression. The residuals are those of the refit of the design of the fit to its response, with its weights and its offset, so on a model with an offset the statistic is that of the model fitted to the response minus the offset, Koenker's statistic of the model (since 0.16.0). The implementation shares the validation helpers used across the package to ensure that: (i) the model satisfies minimum sample-size thresholds via rvalidateModelInputs(), (ii) the response, design, weights and offset are those of the fit itself, with the rows of `data` matched to its observations (the model context, since 0.16.0), and (iii) the requirements registered for this test in rvalidateTestRequirements() are met.

## Value

An object of class `htest` reporting the chi-squared statistic and p-value for the null hypothesis of constant error variance.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. A model with an offset is accepted since 0.16.0: the refit of the design carries the offset, as `performKoenkerTest()` accounts for it. It refuses a multivariate `lm` ("needs a model with one response"), and, since 0.16.0, an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, the basis of its smooths, that the test fits again. Since 0.17.0 a `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is tested on that design and gives the value of the `lm()` of it; before, it was refused as a penalised one. A `gam` without smooth terms is tested as the Gaussian `glm` it is (decisions 37 and 38 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Koenker, R. (1981). A note on studentizing a test for heteroscedasticity. *Journal of Econometrics, 17*(1), 107–112. doi:10.1016/03044076(81)900622

Davidson, R., & MacKinnon, J. G. (2004). *Econometric Theory and Methods*. Oxford University Press. Section 7.5 derives the test as $nR^2$ from the regression of the squared residuals on the variance regressors.

## See Also

`performBPTest()` for the classical LM statistic, which assumes normal errors, and `performKoenkerTest()`, which computes the same studentized $nR^2$ statistic on the squared residuals. The robust workflow in performBPTestRobust() augments the studentized statistic with bootstrap diagnostics.

## Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
# Deprecated: warns, and returns the statistic of performKoenkerTest()
suppressWarnings(performStudentizedBPTest(mod, mtcars))
performKoenkerTest(mod, mtcars)
```

---

# performSzroeterTest *Szroeter test for ordered alternatives*

---

## Description

Detects monotonic changes in variance when observations can be meaningfully ordered—typically by time or another covariate—using the cumulative weighting scheme proposed by Szroeter (1978).

## Usage

```
performSzroeterTest(model, data = NULL, order_by,
  alternative = c("greater", "two.sided", "less"))
```

## Arguments

| | |
|---|---|
| `model` | A fitted stats::lm object representing the mean equation under study. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The residuals and weights are read from the fit itself, and `order_by` from the rows of `data` that hold the observations of the fit. Without `data` it is read from the model frame of `model`, which holds it only when it is a variable of the model under its own name (`x`, not `log(x)`); a model that transforms the variable it is ordered by needs `data`, as does an ordering variable the model does not use. Since 0.16.0, data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |

`order_by` Character scalar naming the column that defines the ordering of observations prior to computing the statistic. The column must be numeric or coercible to an orderable vector.

`alternative` Character scalar specifying the alternative hypothesis: `"greater"` (the default) tests for variance increasing with `order_by`, `"less"` for variance decreasing, and `"two.sided"` for a change in either direction. Szroeter's test targets monotone alternatives, so the one-sided form is the usual choice.

## Details

After ordering the residuals $\hat{e}_{(i)}$ by `order_by`, the test forms the rank-weighted average of the squared residuals

$$h = \frac{\sum_{i=1}^{n} i\, \hat{e}_{(i)}^2}{\sum_{i=1}^{n} \hat{e}_{(i)}^2},$$

which is Szroeter's (1978) class of statistics evaluated at the canonical weights $h_i = i$. Variance that grows with `order_by` shifts weight onto the high ranks and drives $h$ upwards.

Since 0.16.0 the null mean and variance of $h$ are those of the design. Let $M$ be the residual-maker of the design in the order of the test (the rows $\sqrt{w_i}\, x_i$ on a weighted fit), $A = \mathrm{diag}(1, \ldots, n)$, $k$ the rank of the design and $\nu = n - k$. Under normal errors of constant variance (given the weights) the residuals make $h$ a ratio of quadratic forms, $h = \sum_j \lambda_j z_j^2 / \sum_j z_j^2$, with $z_j$ independent standard normal and $\lambda_j$ the $\nu$ eigenvalues of $MAM$ on the residual space. The ratio is independent of its denominator, so its moments are ratios of moments:

$$E\, h = \frac{\mathrm{tr}(MA)}{\nu} = \frac{n(n+1)/2 - \sum_i i\, l_{(i)}}{\nu}, \qquad \mathrm{Var}\, h = \frac{2\,(\nu\, \mathrm{tr}(MAMA) - \mathrm{tr}(MA)^2)}{\nu^2(\nu+2)},$$

with $l_{(i)}$ the leverages in the order of the test and $\mathrm{tr}(MAMA) = \sum_i i^2 - 2\sum_i i^2 l_{(i)} + \|Q_1' A Q_1\|_F^2$, $Q_1$ an orthonormal basis of the design, so that no eigenvalue is computed and the cost is $O(nk^2)$. The test refers

$$Q = \frac{h - E\, h}{\sqrt{\mathrm{Var}\, h}}$$

to the standard normal distribution. The moments are derived here, by the argument that gives those of the Durbin-Watson statistic, and are not taken from Szroeter (1978). Without regressors they are $(n+1)/2$ and $(n^2-1)/(6(n+2))$. Before 0.16.0 the test used $(n+1)/2$ and $(n^2-1)/(6n)$, Szroeter's asymptotic moments, whatever the design: on a weighted fit whose leverage falls along the ordering the mean of $h$ is above $(n+1)/2$ (by about 0.44 in the design of the validation below, at every $n$), and the test rejected too often. The statistic $h$ is unchanged; $Q$ and the p-value change for every fit, on `lm(mpg ~ wt + qsec, mtcars)` ordered by `wt` from -0.6494 and 0.7420 to -0.6626 and 0.7462.

The test needs at least two residual degrees of freedom, $\nu \geq 2$. With one the residuals lie on a single line, $h$ takes the same value whatever the errors and its null variance is 0: since 0.16.0 the test refuses such a fit, for which it returned a value before.

The residuals and weights are those of the fit, and `order_by` is read from the rows of `data` that hold its observations (the model context, since 0.16.0); the package validation helpers check that the ordering variable is present, and ensure sufficient sample size and variability in the squared residuals.

Ties in order_by keep the order of the rows the model was fitted to. Unless the rows were sorted by the response, that order does not depend on the residuals, so the test keeps its level under the null hypothesis; but with a discrete ordering variable the statistic depends on the order of the rows.

runHeteroTests(tests = "szroeter"), and anything else that runs the registered diagnostic, orders the observations by the fitted values of model, so its alternative is variance that rises with the mean. With a single regressor that is the order of the regressor when its slope is positive and the reverse order when it is negative. The fitted values are computed from the coefficients, so observations with the same regressors tie exactly, as they do under a factor or a binary regressor, and the ties are broken as above. Since 0.16.0 they are read from the fit, in its order, and data is matched to the observations of the fit as this function matches it: without data the registered test reads the model frame, and data in the order of the fit whose row numbers are not those of the fit are taken. Before, it matched the fitted values to the row names of data, and refused both ("Missing rows").

## Value

An object of class htest reporting Szroeter's standardised statistic Q, the sample size, the underlying rank-weighted statistic h in estimate, and the p-value from the normal reference distribution, with the null mean and variance of h given the design.

## Validation

The statistic and its null moments are checked against an independent reconstruction from the full residual-maker in 'tests/testthat/test-pass-a-reference.R', and the moments against the eigenvalues of $MAM$ to 1e-13. Releases before 0.7.0 divided the centred statistic by a further $\sqrt{n}$, shrinking $Q$ by a factor of roughly $2/\sqrt{n}$ and leaving the test with essentially no power against any alternative; see 'NEWS.md'.

inst/validation/szroeter-design-null-size.R tested the samples of inst/validation/harrison-mccabe-size-po lm(y ~ x1 + x2) ordered by x1, at 5000 replications per cell. With Gaussian errors the test rejected a true null hypothesis at the 5% level 0.048, 0.050 and 0.049 of the time at 40, 100 and 400 observations, and 0.049, 0.047 and 0.050 on a correctly weighted fit (weights $1/x_1^2$), all inside the release gate's interval, 0.042 to 0.058; with the moments of 0.15.0 it rejected 0.044, 0.048 and 0.049, and on the weighted fit 0.067, 0.056 and 0.055. Its moments agreed with those of 100,000 simulated samples of one design of each size, variances within 1%. The exact null distribution of $h$ given the design, from the eigenvalues by Imhof's (1961) method, gave the same rates within 0.0006, at a cost of $O(n^3)$ per call. Under $t_5$ errors no version holds the level (0.10 to 0.15): the null assumes normal errors, and performGlejserTest() with robust = TRUE is the test to use there.

## Supported models

It accepts an lm and a glm with the Gaussian family and the identity link, unweighted or with positive weights, a fitWLS() fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a glm of another family or link, a survey::svyglm() fit, a fit with a zero weight, a coefficient that is not finite for another reason, and data that lack rows or variables of the fit. It refuses a multivariate lm ("needs a model with one response"), since 0.16.0; before, it stopped with a coercion error. Since 0.17.0 it refuses an mgcv::gam() fit with penalised smooth terms: the null distribution of the test is derived for the least-squares residuals of the design of the fit, the basis of its smooths, which the residuals of a penalised fit are not; before, it compared them with that distribution. A

gam that is the least-squares fit of its design, as one whose smooth terms are all fixed (s(x, fx = TRUE)), is tested on that design, as before, and gives the value of the lm() of it (decisions 37 and 38 of the design of the model context). Since 0.16.0 the observations of a gam are matched to the rows of data by the row names of its model frame. Before, they were matched by position, since the residuals of a gam carry no names, and data with the rows of the fit in another order gave a wrong value. The null moments of a gam are those of its design, with the rank of that design, as performHarrisonMcCabeTest() takes its null distribution. Which model each test accepts, and why, is set out in test_results, under *Supported models*.

## References

Szroeter, J. (1978). A class of parametric tests for heteroscedasticity in linear econometric models. *Econometrica, 46*(6), 1311–1327. doi:10.2307/1913833

Godfrey, L. G. (1988). *Misspecification Tests in Econometrics*. Cambridge University Press. Section 5.4 outlines the Szroeter test.

Imhof, J. P. (1961). Computing the distribution of quadratic forms in normal variables. *Biometrika, 48*(3-4), 419–426. doi:10.1093/biomet/48.34.419

## See Also

performKoenkerTest() for the regressor-based alternative and performGQTest() for split-sample diagnostics on ordered data.

## Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
performSzroeterTest(mod, mtcars, order_by = "wt")

# Detect ordered heteroscedasticity in simulated data with variance increasing
# in an index variable
set.seed(404)
n <- 150
x <- sort(runif(n))
y <- 2 + 0.5 * x + rnorm(n, sd = 0.4 + 0.8 * x)
df <- data.frame(y, x)
performSzroeterTest(lm(y ~ x, data = df), df, order_by = "x")

# Two-sided version when the direction of the change is not known in advance
performSzroeterTest(mod, mtcars, order_by = "wt", alternative = "two.sided")
```

---

performVarianceFormTest

*Test the functional form of the error variance*

---

## Description

Tests whether the error variance follows a stated parametric function of stated variables, $\sigma_i^2 = h(z_i;\gamma)$. Every other test in the package has constant variance as its null hypothesis, so it can say that the variance moves with some variables but not whether an exponential or a power function of them describes it. Here the variance function is the null hypothesis, and rejecting it means that the chosen shape is inadequate.

## Usage

```
performVarianceFormTest(
  model,
  data = NULL,
  var_formula = NULL,
  form = c("exponential", "power"),
  against = "squares"
)
```

## Arguments

| | |
|---|---|
| `model` | A fitted stats::lm object for the mean equation, or a stats::glm with the Gaussian family and the identity link, or the result of `fitWLS()`. In the last case the variance function that was fitted is the one tested, and `var_formula` and `form` must be left at their defaults. |
| `data` | Optional base::data.frame holding the variables named in `var_formula` or `against`. It is needed only when they are not in the model frame. Since 0.16.0 its rows are matched to the observations of the fit as the tests that read `data` match them, data with row names of their own in any order and data that number their rows (a tibble) in the order of the fit, and other data are refused, where numbered data in another order were read by position. A variable of the formulas that neither `data` nor the model frame holds is refused, where it was found in the workspace. |
| `var_formula` | One-sided formula naming the variance regressors $z_i$, for example `~ x`. `NULL`, the default, uses the regressors of `model`. |
| `form` | Character scalar giving the variance function under the null: `"exponential"` (the default) or `"power"`. They are defined in the section on variance functions of `fitWLS()`. |
| `against` | The terms the variance function is tested against. The default, `"squares"`, adds the squares and pairwise products of the variance regressors, on the scale the form uses (logarithms for the power form). A one-sided formula such as `~ w` or `~ I(x^3)` names the added terms instead, which tests the variance function against a variable it leaves out or against a particular departure. |

## Details

The procedure has three steps.

1. The variance function is fitted to the residuals of `model` and the model is refitted by weighted least squares, exactly as `fitWLS()` does.

2. The squared standardized residuals $r_i = \tilde{e}_i^2 / \hat{\sigma}_i^2$ of the weighted fit are formed. If the variance function is right their mean does not depend on anything.
3. $r_i$ is regressed on a constant, on $g_i = \hat{\sigma}_i^{-2}\, \partial \sigma_i^2 / \partial \gamma$, and on the added terms $a_i$. The statistic is the F test that the coefficients on $a_i$ are zero.

For the exponential form $g_i$ is $z_i$, and for the power form it is $\log z_i$. The regression is the score direction of the variance function extended by the added terms, so the test is a studentized Lagrange multiplier test of the stated form against that extension.

Step 3 keeps $g_i$ in the regression for a reason. Estimating $\gamma$ fits the variance along $g_i$, so those directions say little about the form: a Breusch–Pagan or Koenker test of the weighted fit on the same variance regressors rejects a wrong form about as often as the right one. The information is in the terms outside the fitted variance function. Conditioning on $g_i$ is also what makes the null distribution free of the error in $\hat{\gamma}$ to first order, which is the regression-based construction of Wooldridge (1990, 1991).

## Value

An object of class `htest` with the F statistic, its numerator and denominator degrees of freedom, and the p-value for the null hypothesis that the variance function is correctly specified. `estimate` holds the fitted coefficients of the variance function: slopes of the log-variance for the exponential form and exponents for the power form. The element `variance_function` records the form, the variance regressors and the added terms that entered the test.

## Interpretation and caveats

A rejection says that the added terms move the variance beyond what the stated function allows. A non-rejection says only that those terms do not; it does not establish the form. Two functions that are close over the observed range of $z_i$ cannot be told apart, and the test will then accept both: a variance that is linear in a regressor is accepted as exponential or as a power when the regressor varies over a short range. Running the test for several forms is a comparison of fits, and choosing the form with the largest p-value is not a test.

The test judges the shape, not the fitted coefficients. Because it conditions on $g_i$, it does not respond to weights that are off along the variance regressors themselves. A test of constant variance on the weighted fit is sensitive to that, with the reservations about its level described under weighted fits in `performKoenkerTest()`.

The F reference distribution assumes that the standardized errors have a constant fourth moment, the assumption behind the Koenker test. It does not assume normality.

With $q$ variance regressors the default adds up to $q(q+1)/2$ terms. When that is large relative to the sample, name the variance regressors with `var_formula` or the added terms with `against`.

## Validation

No reference implementation exists, so the statistic is checked against a reconstruction from its definition to `1e-8` (`tests/testthat/test-variance-form.R`) and by simulation. Over 24 designs in which the form under test was right (both forms, constant and heteroscedastic variances, 50 to 400 observations, Gaussian and $t_5$ errors, 5000 replications each) the rejection rate at the 5% level ran from 4.2% to 5.7%. Power against a log-variance that is quadratic in a regressor was 98% at 150 observations. Against the other log-linear form it was 22% at 150 observations and 74% at 400. See `inst/validation/README.md`.

## Supported models

It accepts an unweighted `lm`, a `fitWLS()` fit and, since 0.16.0, an unweighted `glm` with the Gaussian family and the identity link and a rank-deficient fit. The Gaussian `glm` is the least-squares fit `lm()`, with the same design and residuals, from which the variance function is estimated and the model refitted, so the test gives the value of the `lm`, and `fitWLS()` refits it as that `lm`; it refused every `glm` before 0.16.0. With the exponential form, the default, the variance regressors of a rank-deficient fit span the same space with or without its aliased columns, and the test gives the value of the same model fitted without them. With the power form they are the logarithms of every column of the design, the aliased ones included, and the logarithm of an aliased column is not a combination of the logarithms of the others, so the value is not that of the model without them; it does not depend on the order of the terms, as the value of the model without them does. It refuses weights that did not come from `fitWLS()`, the prior weights of a Gaussian `glm` among them, a `glm` of another family or link, a `survey::svyglm()` fit, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.16.0 it is built on the model context, and also refuses a multivariate `lm`, a fit whose coefficients, residuals or weights were changed after the fit, and a fit with `model = FALSE` whose model frame, rebuilt from the data it was fitted to, no longer holds its response, because those data changed after the fit: it refitted that frame and returned a value. A fit with `model = FALSE` whose data are gone is tested on `data` that give its response and fitted values. Without `var_formula` or `against`, the test reads nothing from `data`, which it matches as every test does, and refuses numbered data in another order, rows in front of those of the fit and repeated row names, where it gave the value of the fit. Since 0.17.0 it refuses an `mgcv::gam()` fit with penalised smooth terms, which it refitted by weighted least squares on the basis of its smooths without the penalty, and so tested another fit; a `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), is tested on that design, as before (decisions 37 and 38 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Wooldridge, J. M. (1990). A unified approach to robust, regression-based specification tests. *Econometric Theory, 6*(1), 17–43. doi:10.1017/S0266466600004898

Wooldridge, J. M. (1991). On the application of robust, regression-based diagnostics to models of conditional means and conditional variances. *Journal of Econometrics, 47*(1), 5–46. doi:10.1016/03044076(91)90076P

Harvey, A. C. (1976). Estimating regression models with multiplicative heteroscedasticity. *Econometrica, 44*(3), 461–465. doi:10.2307/1913974

Koenker, R. (1981). A note on studentizing a test for heteroscedasticity. *Journal of Econometrics, 17*(1), 107–112. doi:10.1016/03044076(81)900622

Carroll, R. J., & Ruppert, D. (1988). *Transformation and Weighting in Regression*. Chapman & Hall. Chapter 3 treats the estimation and checking of variance functions.

## See Also

`fitWLS()` fits the variance functions tested here. `performHarveyTest()` and `performParkTest()` test constant variance against the exponential and the power form respectively.

## Examples

```
# Standard deviation proportional to x, so the variance is a power of x
set.seed(2026)
n <- 300
sim <- data.frame(x = runif(n, 1, 5))
sim$y <- 1 + 2 * sim$x + sim$x * rnorm(n)
mod <- lm(y ~ x, data = sim)

# The power form is the true one; estimate is the fitted exponent
performVarianceFormTest(mod, form = "power")

# The exponential form in x is not
performVarianceFormTest(mod, form = "exponential")

# Test the variance function of a weighted fit directly
wls <- fitWLS(mod, form = "power")
performVarianceFormTest(wls)

# Test a variance function against one particular added term
performVarianceFormTest(mod, form = "exponential", against = ~ log(x))
```

---

# performVIFDiagnostic *Variance inflation factors*

---

## Description

Compute variance inflation factors for assessing multicollinearity. Each VIF is the corresponding diagonal element of the inverse correlation matrix of the design, so the computation works for models containing factors (each contrast column receives its own VIF).

## Usage

```
performVIFDiagnostic(model)
```

## Arguments

| | |
|---|---|
| model | A fitted `lm` model. |

## Value

A named numeric vector of VIF values, one per design-matrix column (excluding the intercept). Perfectly collinear columns are reported as `Inf`.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performVIFDiagnostic(m)
```

`performWhiteTest` *Perform White's test for heteroscedasticity*

## Description

Implements White's test on a fitted linear model.

## Usage

```
performWhiteTest(model, data = NULL, cross_products = TRUE, max_interactions = 10)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm`. |
| `data` | The data frame `model` was fitted to, or `NULL`, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an `offset()` term, or takes its `weights` or `offset` from a column of the data needs no `data`, and the model frame of `model` is accepted as `data` (since 0.16.0; before, such models needed `data`). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see `test_results`. |
| `cross_products` | Logical. Include cross-product terms in the auxiliary regression? |
| `max_interactions` | Maximum number of cross-product terms admitted to the auxiliary regression. Guards the auxiliary design against growing quadratically with the number of regressors. |

## Details

An auxiliary regression of $e^2$ on all regressors, their squares and cross-products produces $R^2$. The statistic $nR^2$ follows a chi-square distribution whose degrees of freedom are the rank of the auxiliary regression less its intercept (see *Degrees of freedom*).

The test is computed in memory whatever the size of the data. Up to 0.15.0 it switched to `performWhiteTestStreaming()` on data above 100 MB, which until 0.17.0 stopped on any model with a factor or a `poly()` term; call that function to stream.

## Value

An object of class `htest` containing the test statistic, p-value and degrees of freedom.

### Degrees of freedom

Let $Z$ hold the regressors, their squares and their cross-products. The auxiliary regression of $e^2$ on $Z$ has an intercept, and the degrees of freedom are the rank of $[1, Z]$ minus one, as `lmtest::bptest()` counts them given the same terms as `varformula`. A column of $Z$ that is a linear combination of the intercept and the columns before it is dropped: the square of a dummy, which equals the dummy, and, when the columns of $Z$ span the constant, one column more. They span it with the squares of the dummies of `contr.sum`, `contr.helmert` or `contr.poly` contrasts, the last the default for an ordered factor, the dummies of every level of a factor in a model without an intercept, an interaction with an empty cell such as that of the baseline levels, or the squares of an orthogonal `poly()` basis. In 0.15.0 the degrees of freedom were the rank of $Z$ alone, one more on such a design, with the same statistic and a larger p-value: on `lm(y ~ x1 + f, contrasts = list(f = "contr.sum"))`, with a factor of two levels, 5 where `lmtest::bptest()` gives 4, and on `lm(y ~ x1 + o)`, with an ordered factor of four levels, 9 for 8.

The rank is decided, at the default tolerance of `qr()`, on the auxiliary design of the centred regressors, which spans the same columns with the intercept, so that it does not depend on where the regressors are located; the regression is fitted on $Z$, or on that design when the fit of $Z$ would lose one of the columns kept. Before, a regressor far from zero, such as $x = 10^4 + u$ or a date written as `YYYYMMDD`, lost its square, which the rank of $[1, Z]$ took for a combination of the intercept and the regressor: on `lm(y ~ x1 + x2)` with $10^4$ added to `x1`, 10.34 on 4 degrees of freedom, where the model of `x1` as it was gives 11.14 on 5, as both now do. That holds for regressors that enter the model linearly, which are centred one by one. A product or a power of a variable located far from zero inside the model stays ill-conditioned once its columns are centred, and can still lose a column; how far is far depends on the spread of the variable. On 200 observations of two standard normal regressors, `lm(y ~ I(x1 + 2e3) * I(x2 + 2e2))`, with `x1` about 2,000 standard deviations from zero, keeps every coefficient, and White's test gives 46.83 on 7 degrees of freedom where `lm(y ~ x1 * x2)`, the same model, gives 51.90 on 8; `lm(y ~ I(x1 + 2e3) + I((x1 + 2e3)^2))` gives 8.05 on 3 for 11.29 on 4. In ten such data sets, of normal and of uniform regressors, the degrees of freedom held, and the statistic moved by at most 3e-5 relative, while the shift was at most 700 standard deviations of the regressor; the column was lost from 800 to 2,000 standard deviations on, depending on the term and the data. From about 3,000 to 11,000 standard deviations on, `lm()` itself aliases the interaction or the square, before White's test sees it, and the test is that of the model without the term: with `I(x1 + 1e4) * I(x2 + 1e3)`, 35.02 on 5, the value of `lm(y ~ x1 + x2)`. A year from 1990 to 2020, about 220 standard deviations from zero, and its square, `year * z` or `income * age` are far from that, and give the value of the centred model to seven digits or more.

On a design whose auxiliary columns do not span the constant and whose regressors are not located far from zero, such as continuous regressors or factors with the default treatment contrasts, nothing changed. The same holds with `cross_products = FALSE` or more regressors than `max_interactions`, and for the tests computed through this function: `performWhiteTestRobust()`, whose interval, effect size and power follow the degrees of freedom, and the registered `"white"` of `runHeteroTests()`. The p-value of `performWhiteTestBootstrap()` depends on the statistic alone and does not change on a design that spans the constant. `performWhiteTestStreaming()` gives these degrees of freedom, also on such a design, whose auxiliary columns are collinear with its intercept, on which it stopped until 0.17.0, and, since 0.17.0, on an `mgcv::gam()` fit with smooth terms, whose basis it tests, where it tested the variables of the formula. On an auxiliary design whose columns are nearly collinear, as the squares and products of the basis of a smooth term can be, the columns kept can differ from those kept in 0.15.0, so the statistic can change too: there the auxiliary regression of 0.15.0 could silently lose a column to its intercept while its degrees of

freedom still counted it, as on a `mgcv::gam(y ~ s(x1) + x2)` fit whose statistic moves by 0.28%, to the value of `lmtest::bptest()` with the same terms.

### Too few observations

An auxiliary regression that fits the squared residuals exactly has an $R^2$ of 1, and the statistic is the number of observations, whatever the data. It fits them exactly with as many independent terms as observations, and with fewer when the squared residuals are a combination of its terms: on a paired design, such as `lm(y ~ treat + subject)` with two observations of each subject, the two residuals of a subject sum to zero, so their squares are equal and a combination of the subject dummies. On 30 subjects the test gave 60 on 30 degrees of freedom (p = 0.0009) on homoscedastic data. Since 0.16.0 it refuses such a regression, where it returned that statistic, and says what to use, as the call allows: `cross_products = FALSE` when the products were in and the regressors alone do not fit the squared residuals exactly, Koenker's test when they do not, and more data when the regression has as many terms as observations. Six continuous regressors have 27 squares and products, 28 terms with the intercept, so they need at least 29 observations. With fewer terms than observations the design forces the exact fit, and more data of the same design do not help: 500 subjects of two observations are refused as 30 are, and the refusal asks for more observations per level.

A fit counts as exact when its residual sum of squares is at most `sqrt(.Machine$double.eps)`, about $1.5 \times 10^{-8}$, of the total sum of squares of the squared residuals. An exact fit leaves rounding error, about $10^{-30}$ of the total on the paired design; squared residuals that depart from the fit by $2 \times 10^{-4}$ of their size leave about $2 \times 10^{-7}$, and are tested.

The robust form, the registered `"white"`, `performWhiteTestBootstrap()` and `performWhiteTestStreaming()` refuse both kinds of exact fit too. `runHeteroTests()` falls back on White's test without cross products, as on any refusal of White's test, unless the regressors alone fit the squared residuals exactly, as on the paired design, or that is not known, as in the streaming test: the Breusch-Pagan and Koenker tests, its other fallbacks, regress the squared residuals on the regressors and would return the number of observations too, so the runner stops with the refusal.

The refusal covers the exact fit only, and one step away from it the test still rejects almost always: in 2,000 homoscedastic draws of `lm(y ~ x + subject)`, with normal `x` and errors, White's test rejected at the 5% level in 99% of them with 30 subjects of two observations and one of them observed a third time, and in all but one with 30 pairs whose weights vary within pairs. Those are fixed effects for small groups, on which the test warns (see *Fixed effects for small groups*).

### Fixed effects for small groups

White's, Koenker's and the Breusch-Pagan tests are not valid with fixed effects for small groups: a factor with many levels and few observations of each, such as `subject` in `lm(y ~ x + subject)` with each subject observed a few times. On homoscedastic data they reject too often, and the more often the more levels the factor has and the fewer observations each level has. The excess comes from the dummies of the factor in the auxiliary regression. The residuals of a level are taken from its own mean, so under normal errors the squared residuals of a level of $T$ observations vary between levels $T/(T-1)$ times as much as the auxiliary regression assumes, and with $G$ levels the statistic exceeds its degrees of freedom by about $(G-1)/(T-1)$. In the study below Koenker's statistic had that mean, and the squared residuals regressed on `x` alone, without the dummies, were rejected in 4.6% to 5.5% of the draws by Koenker's statistic and in 4.3% to 5.6% by the Breusch-Pagan statistic (`inst/validation/fixed-effects-dummies.R`). The excess grows with $G$ at a fixed $T$, while the

spread of the chi-squared distribution grows as $\sqrt{2G}$, so with a fixed number of observations per level the size goes to 1 as the number of levels grows.

Since 0.16.0 the test warns when a term of the model that holds a factor or a character variable has more levels than observations per level: $L > n/L$, that is $L^2 > n$, with $L$ the number of distinct combinations of the factors of the term among the $n$ observations of the fit. The term is a factor, such as `subject`; an interaction of factors, counted by its cells, such as `a:b` of a crossed `a * b` or `school:class` of a nested `school / class`; or a slope for each level, `x:subject`. A variable left out of the formula, as in `y ~ . - id`, does not count. The warning names the term and gives $L$, $n$ and the mean number of observations per level; the statistic, the degrees of freedom and the p-value are those the test computes without it. It has class `heteroTests_fixed_effects_warning`, which `suppressWarnings(expr, classes = "heteroTests_fixed_effects_warning")` muffles, and no other warning. It is given on every call, also when `cachedTest()` or `runHeteroTests()` answers from its cache, and `runHeteroTests()` and `runDiagnostics()` give it once for each test that gives it. `performWhiteTestRobust()` and the registered `"white"` give it too, and so does `performWhiteTestStreaming()`, which since 0.17.0 computes on a factor that is a main effect, whose dummies and their squares are collinear, where it stopped, as well as on a slope for each level. `performWhiteTestBootstrap()`, whose p-value is a bootstrap's, held its level on these designs and does not give it.

On a balanced design of $G$ subjects of $T$ observations the rule is $G \leq T$. In the study it kept the three tests within 2.5% to 7.5% at the 5% level where it holds, and put at least one of them outside where it does not, in 68 of the 69 designs where they computed; the exception is the Breusch-Pagan test on 20 subjects of 20, at 8.0%. Unbalanced groups, a second regressor and a regressor correlated with the subject effects followed the same pattern, with the mean number of observations per level, $n/G$, in place of $T$. The rule is a warning, not a correction, and many panels fail it: on 100 firms observed over 10 years the three tests reject in 18% to 21% of the draws. Rejection at the 5% level, in percent, in 2,000 homoscedastic draws of `lm(y ~ x + subject)`, with `x`, the subject effects and the errors standard normal:

| Subjects | Observations of each | White | Koenker | Breusch-Pagan | Warns |
|---|---|---|---|---|---|
| 10 | 10 | 6.4 | 6.2 | 7.3 | no |
| 20 | 20 | 6.0 | 6.4 | 8.0 | no |
| 5 | 4 | 1.0 | 5.4 | 7.0 | yes |
| 30 | 3 | 50.7 | 57.3 | 39.6 | yes |
| 30 | 4 | 22.2 | 23.9 | 28.7 | yes |
| 100 | 8 | 23.8 | 23.8 | 25.2 | yes |
| 100 | 10 | 18.1 | 18.4 | 21.3 | yes |
| 200 | 4 | 97.8 | 97.6 | 82.5 | yes |

A slope for each level counts too, but there the error goes the other way: with a slope for each subject, `lm(y ~ x + x:subject)`, on 30 subjects of 3 and of 4 and on 100 of 8, the three tests rejected too rarely, White's in 0.7% to 2.2% of the draws, Koenker's in 0.1% to 0.8% and the Breusch-Pagan test in 0.4% to 2.8% (`inst/validation/fixed-effects-variants-size.R`). The warning therefore says that the p-value is unreliable: in the simulations usually far too small, and too large with a slope per level.

With at most `max_interactions` regressors White's auxiliary regression holds the products of `x` with the dummies, a slope for each subject, and on 5 subjects of 4 it rejected too rarely, in 1.0%. The

study is `inst/validation/fixed-effects-size.R`, with its results in `fixed-effects-size.csv` (`system.file("validation", package = "heteroTests")`). Juhl and Sosa-Escudero (2014) derive a test of heteroskedasticity for this setting, after fixed effects estimation of a panel of many units observed a fixed number of times; it needs to be told which factor is the unit, and the package does not implement it.

### Weighted fits

On a fit with weights the test is computed from the Pearson residuals $\sqrt{w_i}\, e_i$, so it tests whether the weights are adequate; see the section on weighted fits in `performKoenkerTest()`, which also covers weights estimated by `fitWLS()`.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4), 817–838. doi:10.2307/1912934

Greene, W. H. (2018). *Econometric Analysis* (8th ed.). Pearson.

Juhl, T., & Sosa-Escudero, W. (2014). Testing for heteroskedasticity in fixed effects models. *Journal of Econometrics*, 178(Part 3), 484–494. doi:10.1016/j.jeconom.2013.07.005

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
performWhiteTest(m, mtcars)
```

---

## performWhiteTestBootstrap

*Bootstrap White test*

---

### Description

Approximates the finite-sample distribution of White's LM statistic by resampling the fitted residuals, refitting the model, and recalculating the test statistic across many bootstrap replications.

## Usage

```
performWhiteTestBootstrap(
  model,
  data = NULL,
  B = 999,
  parallel = FALSE,
  n_cores = NULL,
  reestimate_weights = NULL
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object representing the mean specification to be diagnosed. |
| data | The data frame model was fitted to, or NULL, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.17.0; before, such models needed data). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| B | Integer scalar giving the number of bootstrap replications. Defaults to 999. The p-value is (1 + #) / (B_eff + 1), where B_eff is the number of replicates whose statistic is finite (B when every replicate succeeds), so a test at level alpha is exact only when alpha * (B_eff + 1) is an integer: with 999 it is at the 5% and 1% levels, and with 1000, the default before 0.15.0, at neither. |
| parallel | Logical scalar; if TRUE the replicates are computed in parallel, by forked workers (parallel::mclapply()) where the platform can fork and on a socket cluster (parallel::makeCluster()) elsewhere. See *Parallel replicates*. |
| n_cores | Optional integer number of worker processes when parallel = TRUE. Defaults to parallel::detectCores() - 1. The result does not depend on it. |
| reestimate_weights | Whether each replicate estimates the weights of a fitWLS() fit again from the variance function fitWLS() recorded on it. NULL, the default, does so for a fitWLS() fit and holds the weights fixed for any other model. FALSE holds them fixed, as every call did before 0.15.0. TRUE is an error for a model that fitWLS() did not return, or on which it fell back to equal weights. Each replicate regenerates the response and holds the design fixed, so a variance regressor that uses a variable of the response which the right-hand side of the model does not use, such as y or x for I(y / x) ~ z, or that is the response term itself, as ~ . gives, cannot be estimated again: NULL then holds the weights fixed, with a warning, and TRUE is an error. Giving the response as a column of the data, as in r ~ z, says which variables stay. See *Weighted fits*. |

## Details

For a fitted model $\hat{y} = X\hat{\beta}$, the algorithm proceeds as follows:

1. Compute White's LM statistic $nR^2$ from the auxiliary regression on squares and cross-products of the regressors.
2. Generate $B$ bootstrap samples by resampling the centred residuals with replacement, forming $y^{*(b)} = \hat{y} + \hat{e}^{*(b)}$.
3. Refit the model to each bootstrap sample and recompute White's statistic $T^{*(b)}$ using the same auxiliary specification.
4. Estimate the bootstrap p-value as $\hat{p} = (1 + \sum_b I\{T^{*(b)} \geq T_{\text{obs}}\})/(B_{\text{eff}} + 1)$, where $b$ runs over the $B_{\text{eff}}$ replicates whose statistic is finite, so the p-value is never zero.

The bootstrap distribution offers improved size control for moderate sample sizes or high-dimensional designs where the chi-squared approximation may be inaccurate. When `parallel = TRUE` the resampling step exploits available CPU cores to reduce computation time.

## Value

A `htest` object reporting both the asymptotic White statistic and its bootstrap p-value. The returned object also stores the simulated test statistics in `boot_statistics` for further inspection.

## Parallel replicates

With `parallel = TRUE` replicate `i` draws its random numbers from stream `i` of a L'Ecuyer-CMRG generator (`parallel::nextRNGStream()`), seeded by one number drawn from the session's generator. A call made after `set.seed()` therefore returns the same result each time, whatever the number of workers and whether they are forked or run on a socket cluster. It does not return the result of the same call with `parallel = FALSE`, which draws every replicate from the session's generator, as it did before 0.15.0: the two are bootstraps of the same data with different draws, and each is valid. Before 0.15.0 a seeded call with `parallel = TRUE` gave a different result each time, and on Windows more than one worker was an error.

## Weighted fits

A fit with weights $w_i$ is ordinary least squares on $\sqrt{w_i}\,y_i$ and $\sqrt{w_i}\,x_i$, whose errors have constant variance under the null hypothesis, and the bootstrap is carried out in that regression. What is resampled is its residuals, the Pearson residuals $\sqrt{w_i}\,e_i$. When the model has an intercept they are centred, as the residuals of an unweighted fit with an intercept are, because the weighted regression has no constant column. Each draw is divided by $\sqrt{w_i}$ to return it to the scale of the observation it is added to, and each replicate is refitted with the weights, so `performWhiteTest()` computes its statistic from the Pearson residuals as it does for the fit supplied. With equal weights the result is that of the unweighted fit.

Weighted or not, each replicate is refitted on the stored design of the fit with its total offset, the `offset` argument and the `offset()` terms (see *The design of the replicates*). A transformed response, such as `log(y)` in `log(y) ~ x`, is regenerated on the scale the model uses.

Weights given to the fit are known, and they are held fixed in every replicate. On a fit returned by `fitWLS()` they were estimated from the same data, and since 0.15.0 they are estimated again in each replicate by default (`reestimate_weights`): the replicate is fitted as the model `fitWLS()`

was given, the variance function `fitWLS()` recorded is fitted again to its residuals, on the same regressors and with the same form, and the replicate is refitted with the weights it gives, so that `performWhiteTest()` sees a `fitWLS()` fit of the replicate. A replicate whose variance function cannot be fitted, where `fitWLS()` would fall back to equal weights, counts as a failed replicate, and the p-value is computed from the replicates that remain; a warning gives the number that failed.

In `inst/validation/weighted-fits-size.R`, the data have standard deviation $\exp(0.3x)$, the variance function is correctly specified, the errors are normal, there are 150 observations and `B = 499`. At the 5% level the test then rejected 0.064 of the time when it held the weights of `fitWLS()` fixed, as every call did before 0.15.0 and `reestimate_weights = FALSE` still does. It rejected 0.0415 when it estimated them again, just below the release gate's interval, 0.042 to 0.058, and 0.054 with the true weights. The section on weighted fits in `performWildBootstrapTest()` has the larger error of Koenker's statistic.

## The design of the replicates

Since 0.17.0 the observed statistic and every replicate are computed on the design stored with the fit, the design `performWhiteTest()` reads: each replicate, the fitted values plus the errors drawn, is refitted by least squares on that design, with the weights and the total offset of the fit, and its statistic is White's on the same design, without the columns its refit aliased. Nothing is evaluated again from `data` or the formula. Up to 0.16.0 both were computed on a design rebuilt from the rows of `data`, each replicate refitted from the formula: `cut(x, quantile(x))` computed other breaks on the rows of the fit and stopped the test ("factor ... has new levels"); a term computed from all the rows of the data, such as `I(x > median(x))`, took other values on a fit that dropped rows; a fit whose weights or offset are a column of the data, or with an `offset()` term or a transformed variable, stopped without `data`; and an `mgcv::gam()` fit, whose residuals carry no names, given its data in another order, had each draw added to the fitted value of another row. The random numbers are drawn as before, in the same order, so a call made after `set.seed()` returns what it returned wherever the stored design is the design rebuilt from `data`.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.17.0 it refuses an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, the basis of its smooths, that each replicate fits again (before, the refit from the formula stopped on the smooth term), and it tests a `gam` that is that fit, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`), on that design, with the value and the replicates of the `lm()` of it; a `gam` without smooth terms is tested as the Gaussian `glm` it is (decisions 37 and 38 of the design of the model context). It also refuses a fit whose coefficients, residuals or weights were changed after it was fitted, from which it computed the value of the fit before the edit, and a `fitWLS()` fit of a `gam` with smooth terms, whose terms do not rebuild the design of its coefficients, on which it computed a value on the variables of the formula. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

Davidson, R., & MacKinnon, J. G. (2006). The power of bootstrap and asymptotic tests. *Journal of Econometrics, 133*(2), 421–441. doi:10.1016/j.jeconom.2005.02.002

## See Also

`performWhiteTest()` for the classical statistic and performWhiteTestRobust() for enriched diagnostics.

## Examples

```
# The bootstrap needs at least 50 observations, so mtcars (32) is too small.
set.seed(42)
n <- 120
sim <- data.frame(x = runif(n, 1, 5))
sim$y <- 1 + 2 * sim$x + rnorm(n, sd = 0.3 + 0.6 * sim$x)
mod <- lm(y ~ x, data = sim)
performWhiteTestBootstrap(mod, sim, B = 199)


# More replications give a smoother p-value estimate
performWhiteTestBootstrap(mod, sim, B = 499)

# parallel = TRUE runs the replicates on n_cores workers, forked where the
# platform can fork and on a socket cluster elsewhere. Two is the most
# R CMD check --as-cran allows, and the result does not depend on it.
set.seed(42)
performWhiteTestBootstrap(mod, sim, B = 199, parallel = TRUE, n_cores = 2)
```

---

`performWhiteTestRobust`

*Robust White test with bootstrap and effect sizes*

---

## Description

Extends `performWhiteTest()` with optional bootstrap resampling, confidence intervals, effect size reporting, and power analysis.

## Usage

```
performWhiteTestRobust(
  model,
  data = NULL,
  method = c("standard", "reduced"),
  bootstrap = FALSE,
```

```
  B = 999,
  ci_level = 0.95,
  parallel = FALSE
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object representing the mean specification to be diagnosed. |
| data | The data frame model was fitted to, or NULL, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.17.0; before, such models needed data). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| method | Character string selecting the auxiliary specification. "standard" retains squares and cross-products, while "reduced" excludes cross-products for high-dimensional designs. |
| bootstrap | Logical, compute bootstrap diagnostics for the statistic and p-value? The bootstrap is bootstrapTestStatistic(), so on a fit returned by fitWLS() each replicate estimates the weights again, its default since 0.15.0. There is no argument to hold them fixed here; bootstrapTestStatistic() with reestimate_weights = FALSE does that. |
| B | Number of bootstrap replications when bootstrap = TRUE. Defaults to 999, at which a bootstrap p-value of the form (1 + #) / (B + 1) gives an exact test at the 5% and 1% levels; it was 1000 before 0.15.0. The value is recorded in robust_details$bootstrap$B whether or not the bootstrap runs. |
| ci_level | Confidence level for reported intervals. |
| parallel | Logical, allow parallel bootstrap evaluation when the parallel package is available. A seeded call gives the same result each time; see *Parallel replicates* in bootstrapTestStatistic(). |

## Details

Provides enriched inference around the White test by combining bootstrap resampling (Efron & Tibshirani, 1993) with asymptotic approximations. The robust_details element summarises interval estimates, effect sizes, and power calculations to aid decision-making.

## Value

An object of class htest augmented with a robust_details list containing bootstrap, effect size, and power information.

### Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. An `mgcv::gam()` fit with penalised smooth terms is tested on its design, the basis of its smooths, as by `performWhiteTest()`, and refused with `bootstrap = TRUE` (since 0.17.0; before, the bootstrap refused every `gam` with an error of its own): each replicate fits that design again by least squares, which a penalised fit is not. A `gam` that is the least-squares fit of its design, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`) or one without smooth terms, is bootstrapped too, as the `lm()` of that design (decisions 37 and 38 of the design of the model context). Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

### References

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroscedasticity. *Econometrica, 48*(4), 817–838.

Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall.

### See Also

`performWhiteTest()` for the base statistic and `performWhiteTestBootstrap()` for a lighter-weight resampling option.

### Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
performWhiteTestRobust(mod, mtcars, bootstrap = TRUE, B = 200)
```

---

## `performWhiteTestStreaming` *Memory-efficient White test for large datasets*

---

### Description

Computes White's statistic chunk by chunk rather than fitting the full auxiliary regression in memory. This streaming approach allows the test to scale to datasets that would otherwise exhaust available RAM.

## Usage

```
performWhiteTestStreaming(
  model,
  data = NULL,
  chunk_size = 10000,
  cross_products = TRUE,
  max_interactions = 10,
  progress = interactive()
)
```

## Arguments

| | |
|---|---|
| model | A fitted stats::lm object representing the mean specification to be diagnosed. |
| data | The data frame model was fitted to, or NULL, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data needs no data, and the model frame of model is accepted as data (since 0.16.0; before, such models needed data). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. |
| chunk_size | Positive integer specifying the number of observations per chunk. Smaller values reduce memory usage at the expense of additional iteration overhead. |
| cross_products | Logical scalar indicating whether to include all pairwise cross-products of the regressors in the auxiliary regression. Defaults to TRUE and should remain enabled unless dimensionality makes the regression unstable. |
| max_interactions | Single positive integer giving the maximum number of original predictors for which cross-products are generated. When the number of regressors exceeds this threshold, cross-products are dropped to avoid explosive growth in columns. Defaults to 10. |
| progress | Logical flag indicating whether a progress bar should be displayed while streaming the data. Defaults to interactive(). |

## Details

The streaming implementation builds the auxiliary design, the regressors, their squares and their cross-products, chunk by chunk, without materialising it whole. Each chunk is reduced by a QR decomposition stacked on the triangular factor of the chunks before it, which holds the cross-products of every row without forming them. The rank of the auxiliary regression and its $nR^2$ statistic are computed from that factor as in performWhiteTest(), ensuring numerical equivalence while reducing peak memory usage: the regressors are centred at their means, the rank is that of the auxiliary design at the default tolerance of qr(), and a term that is a combination of the intercept and the terms before it is dropped (see *Degrees of freedom* in performWhiteTest()).

Since 0.17.0 the test drops such terms: the square of a dummy variable, which is the dummy, that of the first column of an orthogonal `poly()` basis, or that of `x` beside `I(x^2)`, as `performWhiteTest()` drops them, and gives its degrees of freedom: 17.83 on 8 on `lm(y ~ x1 + group)` with a factor of four levels. Up to 0.16.0 it accumulated the cross-products of the uncentred auxiliary design, decided their rank by `qr()` of them, which squares the condition of the design, and stopped on such terms ("Auxiliary regression became rank deficient"), so on any model with a factor, and on terms that only looked collinear in those cross-products: the squares of a regressor 100 standard deviations from zero, or the terms of `lm(y ~ x1 * x2)` with `x1` between 0.5 and 2. It took the squared residuals for constant when their total sum of squares about their mean was at most `.Machine$double.eps`, whatever their scale, and so refused residuals of about $10^{-5}$: on 120 observations of such residuals it now gives 12.787 on 5 degrees of freedom, the value of `performWhiteTest()`. It refuses squared residuals whose total sum of squares is at most `.Machine$double.eps` of their sum of squares, which agree to about eight digits. And in more than one chunk, on a regressor `a` beside a regressor `a_sq`, it put the square of `a`, which it names `a_sq` too, in place of that of `a_sq` in every chunk after the first: 27.76 in chunks of 7 where `performWhiteTest()` gives 27.64. It no longer uses the sparse cross-products of `Matrix` for large chunks.

Since 0.16.0 the chunks are rows of the design of the fit, read from the fit itself as `performWhiteTest()` reads it, so the value does not depend on the size of the chunks but by rounding; the auxiliary design, several times wider, is built chunk by chunk. Up to 0.16.0 each chunk built its own design from the formula, on its rows of `data`, so a term computed from the rows it is given took other values in each chunk than in the fit: `I(x > median(x))` the median of the chunk, and `cut(x, quantile(x))` its breaks, which stopped the test ("factor has new levels") in chunks of any size on a fit that dropped the row at the minimum of `x`. Since 0.17.0 an `mgcv::gam()` fit with smooth terms, penalised or fixed, is tested on its design, the basis of its smooths, as by the exact tests, read chunk by chunk, and gives their value; before, it was tested on the design of the variables of its formula, which its model frame holds (decision 37 of the design of the model context).

## Value

A `htest` object containing the chi-squared statistic, p-value, and metadata about the chunked computation.

## Supported models

It accepts an `lm` and, since 0.16.0, a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. The Gaussian `glm` with prior weights `w` is the least-squares fit `lm(..., weights = w)`, with the same design and residuals, and the test gives the value of the `lm`. It refused every `glm` before 0.16.0, because the checks it runs read the R-squared of `summary()`, which a `glm` does not report; they read that of the least-squares fit now. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, `data` that lack rows or variables of the fit, and, since 0.16.0, a multivariate `lm`, a fit changed after it was fitted and a fit with `model = FALSE` whose data are gone, unless `data` gives its response and fitted values (see `test_results`). Since 0.17.0 it drops the terms of its auxiliary regression that are collinear, as the squares of dummy variables are, as `performWhiteTest()` drops them, where it stopped. Since 0.16.0 it stops with the refusal of `performWhiteTest()` (see its section *Too few observations*) when its independent terms are at least as many as the observations, where it stopped

as on collinear terms, or fit the squared residuals exactly, collinear or not, as on a paired design. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroscedasticity. *Econometrica, 48*(4), 817–838.

## See Also

`performWhiteTest()` for the exact computation and performWhiteTestRobust() for enhanced reporting.

## Examples

```
data(mtcars)
performWhiteTestStreaming(lm(mpg ~ wt + qsec, data = mtcars), mtcars, chunk_size = 16)
```

---

# performWildBootstrapTest

*Wild bootstrap heteroscedasticity test*

---

## Description

Calibrates the Breusch-Pagan Lagrange multiplier statistic with a *null-imposed* wild bootstrap, providing a p-value that is accurate in small samples and under non-normal errors without relying on the asymptotic $\chi^2$ approximation.

## Usage

```
performWildBootstrapTest(
  model,
  data = NULL,
  B = 999,
  distribution = c("rademacher", "mammen"),
  progress = interactive(),
  reestimate_weights = NULL
)
```

## Arguments

| | |
|---|---|
| `model` | A fitted stats::lm object describing the mean structure whose residual variance is to be assessed. |

`data` The data frame `model` was fitted to, or `NULL`, the default. The response, design, weights and offset are read from the fit itself, so a model that transforms its variables, has an `offset()` term, or takes its `weights` or `offset` from a column of the data needs no `data`, and the model frame of `model` is accepted as `data` (since 0.17.0; before, such models needed `data`). Data that are given are matched to the observations of the fit: rows with row names of their own by name, in any order; rows that only number themselves (a tibble, a data.table, a data frame whose row names were reset) by the row numbers of the fit, or in the order of the fit when they hold one row per observation. The rows the fit dropped may be there or not. Data whose rows cannot be shown to hold the observations of the fit are refused; see test_results. The test reads nothing else from `data`: its replicates are refitted on the stored design.

`B` Integer number of bootstrap replications. Defaults to `999`, which is the default of every resampling test in the package since 0.15.0; it was `499` before. The p-value is `(1 + #) / (B_eff + 1)`, so a test at level `alpha` is exact only when `alpha * (B + 1)` is an integer, which 999 is at the 5% and 1% levels.

`distribution` Multiplier distribution used for the wild perturbation. Supported options are "rademacher" and "mammen".

`progress` Logical flag controlling whether progress should be reported when running the bootstrap loop.

`reestimate_weights`
Whether each replicate estimates the weights of a `fitWLS()` fit again from the variance function `fitWLS()` recorded on it. `NULL`, the default, does so for a `fitWLS()` fit and holds the weights fixed for any other model. `FALSE` holds them fixed, as every call did before 0.15.0. `TRUE` is an error for a model that `fitWLS()` did not return, or on which it fell back to equal weights. Each replicate regenerates the response and holds the design fixed, so a variance regressor that uses a variable of the response which the right-hand side of the model does not use, such as `y` or `x` for `I(y / x) ~ z`, or that is the response term itself, as `~ .` gives, cannot be estimated again: `NULL` then holds the weights fixed, with a warning, and `TRUE` is an error. Giving the response as a column of the data, as in `r ~ z`, says which variables stay. See *Weighted fits*.

## Details

The observed statistic is the usual $nR^2$ from the auxiliary regression of the squared residuals on the regressors. The bootstrap reference distribution, however, must be generated **under the homoscedastic null** — otherwise it inherits the heteroscedasticity the test is looking for and the procedure loses all power. Each bootstrap sample is therefore built from leverage-standardised, mean-centred residuals $r_i = e_i/\sqrt{1-h_{ii}}$ (which have a common variance under $H_0$) resampled i.i.d. and perturbed by a wild multiplier $v_i$, so that $y_i^* = \hat{y}_i + r_i^* v_i$. The statistic is recomputed on each refit and the p-value is $(1 + \#\{T_b^* \geq T\})/(B+1)$. The test holds its level with Gaussian and heavy-tailed $t_5$ errors, and with skewed errors only from about 200 observations; see *Size*. Under the alternative the observed statistic is extreme relative to the homoscedastic reference, giving power.

Each replicate is refitted on the design matrix of the fit, with its offset, whether that was given as an argument or written in the formula. The formula is not evaluated again, so a transformed regressor such as `log(x)`, or a transformed response, is refitted on the scale of the fit.

## Value

An object of class `htest` containing the observed Breusch-Pagan statistic, bootstrap p-value, and additional details under the `bootstrap` element.

## Size

In `inst/validation/wild-multiplier-size.R` and `inst/validation/wild-no-multiplier-size.R`, with `B = 999`, 5000 replications per cell and the design `lm(y ~ x1 + x2)` with `x1` uniform on (1, 5) and `x2` standard normal, the test rejected a true null hypothesis at the 5% level as often as this. With `B = 999` a test whose level is exact rejects 4.9% of the time, and the release gate's interval is 0.042 to 0.058.

| Errors | Multiplier | n = 30 | n = 50 | n = 100 | n = 200 |
|---|---|---|---|---|---|
| Gaussian | Rademacher | 0.0442 | 0.0476 | 0.0482 | 0.0448 |
| Gaussian | Mammen | 0.0430 | 0.0508 | 0.0488 | 0.0470 |
| $t_5$ | Rademacher | 0.0472 | 0.0444 | 0.0492 | 0.0438 |
| $t_5$ | Mammen | 0.0460 | 0.0454 | 0.0476 | 0.0460 |
| centred exponential | Rademacher | 0.0820 | 0.0734 | 0.0608 | 0.0526 |
| centred exponential | Mammen | 0.0782 | 0.0698 | 0.0586 | 0.0534 |

With skewed errors, `rexp(n) - 1`, either multiplier needs about 200 observations to hold the level: both are outside the gate at 30, 50 and 100 and inside it at 200. The excess comes from the multiplier: without it the rate falls, by 1.7 points at 30 and 50 observations. Plausibly, the Rademacher multiplier makes the bootstrap errors symmetric, so the reference distribution loses the skewness of the residuals, on which the squared residuals of the refit depend, and the Mammen multiplier keeps their third moment but doubles their fourth. A plain residual bootstrap, the same resampling without a multiplier ($v_i = 1$), which already imposes the null hypothesis, rejected 0.0648, 0.0562, 0.0482 and 0.0466 of the time with the same errors at 30, 50, 100 and 200 observations, holding the level from 50, and 0.042 to 0.047 with Gaussian and $t_5$ errors. It costs a little power in small samples: against $\sigma_i^2 = \exp(0.4 x_{1i})$ with Gaussian errors it rejected 0.180, 0.354 and 0.744 of the time at 30, 50 and 100 observations, where the Rademacher multiplier rejected 0.194, 0.361 and 0.745, about 1.4 points less at 30. It will be available as `distribution = "none"` in 1.0.0, with `"rademacher"` kept as the default.

## Weighted fits

A fit with weights $w_i$ is ordinary least squares on $\sqrt{w_i}\, y_i$ and $\sqrt{w_i}\, x_i$, whose errors have constant variance under the null hypothesis, and the bootstrap works in that regression. The statistic is computed from its residuals, the Pearson residuals $\sqrt{w_i}\, e_i$, with the auxiliary regressors on their original scale; the residuals that are resampled are the Pearson residuals divided by $\sqrt{1 - h_{ii}}$, with the leverages of that regression, and centred; and each replicate refits it. With equal weights the result is that of the unweighted fit.

Weights given to the fit are known, and they are held fixed in every replicate. On a fit returned by `fitWLS()` they were estimated from the same data, and since 0.15.0 they are estimated again in each replicate by default (`reestimate_weights`). The replicate is fitted as the model `fitWLS()` was given: by least squares, with the weights that model had, if any. The variance function `fitWLS()` recorded, with its form, is then fitted again to the residuals, on the regressors `fitWLS()` evaluated,

and the replicate is refitted with the weights it gives. The statistic is computed from the Pearson residuals of that refit, so the reference distribution includes the estimation the observed statistic went through. A replicate whose variance function cannot be fitted, where `fitWLS()` would fall back to equal weights, counts as a failed replicate: the p-value is computed from the replicates that remain, and a warning gives the number that failed.

Holding the weights of a `fitWLS()` fit fixed, as every call did before 0.15.0 and `reestimate_weights = FALSE` still does, leaves the estimation out of the replicates, and the test inherits the error of the tests that take the weights as known. In `inst/validation/weighted-fits-size.R`, the data have standard deviation $\exp(0.3x)$, the variance function is correctly specified, the errors are normal and `B = 499`. At the 5% level the test then rejected 0.086 of the time with 150 observations and 0.105 with 600 when it held the weights fixed, and 0.0605 and 0.0515 when it estimated them again. With the true weights it rejected 0.054 and 0.048. At 2000 replications 0.0605 is 2.2 standard errors above 0.05, and outside the release gate's interval, 0.042 to 0.058. `performVarianceFormTest()` allows for the estimation too.

## Supported models

It accepts an `lm` and a `glm` with the Gaussian family and the identity link, unweighted or with positive weights, a `fitWLS()` fit and, since 0.16.0, a rank-deficient fit, which it tests on the estimable part of its design: it gives the value of the same model fitted without the aliased columns. It refuses a `glm` of another family or link, a `survey::svyglm()` fit, a fit with a zero weight, a coefficient that is not finite for another reason, and `data` that lack rows or variables of the fit. Since 0.17.0 it refuses a fit whose coefficients, residuals or weights were changed after it was fitted, from which it computed the value of the fit before the edit, and a `fitWLS()` fit of an `mgcv::gam()` fit with smooth terms, whose terms do not rebuild the design of its coefficients, on which it computed a value on the variables of the formula. Since 0.17.0 it refuses an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, the basis of its smooths, that each replicate fits again, and it tests a `gam` that is that fit, as one whose smooth terms are all fixed (`s(x, fx = TRUE)`) or one without smooth terms, on that design, its leverages those of the design, with the value and the replicates of the `lm()` of it (decisions 37 and 38 of the design of the model context). Before, it stopped on every `gam` ("lm object does not have a proper 'qr' component"), reading its leverages with `hatvalues()`. Which model each test accepts, and why, is set out in `test_results`, under *Supported models*.

## References

Davidson, R., & Flachaire, E. (2008). The wild bootstrap, tamed at last. *Journal of Econometrics, 146*(1), 162–169.

Godfrey, L. G. (2006). Tests for regression models with heteroskedasticity of unknown form. *Computational Statistics & Data Analysis, 50*(10), 2715–2733.

Wu, C. F. J. (1986). Jackknife, bootstrap and other resampling methods in regression analysis. *The Annals of Statistics, 14*(4), 1261–1295.

## Examples

```
data(mtcars)
model <- lm(mpg ~ wt + qsec, data = mtcars)
if (interactive()) {
  set.seed(123)
  performWildBootstrapTest(model, mtcars, B = 199, progress = FALSE)
```

```
}
```

---

`plot.HeteroDiagnostic` *Plot diagnostics*

---

## Description

Produce basic residual diagnostic plots.

## Usage

```
## S3 method for class 'HeteroDiagnostic'
plot(x, plots = c("residuals_fitted", "spread_level",
  "density", "qq", "bubble_variance"), ...)
```

## Arguments

| | |
|---|---|
| `x` | A `HeteroDiagnostic` object. |
| `plots` | Character vector selecting which diagnostic panels to draw. |
| `...` | Further arguments passed to the underlying plotting routines. |

## Value

List of ggplot objects.

---

`plot.power_analysis` *Plot method for power analysis results*

---

## Description

Plot method for power analysis results

## Usage

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

## Arguments

| | |
|---|---|
| `x` | Object returned by `simulate_power()`. |
| `...` | Further arguments passed to the underlying plotting routines. |

`plotBeforeAfter` *Compare residuals before and after remediation*

## Description

Draws the residuals of two models against their fitted values in two panels, side by side, to show what a remediation method (a weighted fit, a transformation) did to the spread.

## Usage

```
plotBeforeAfter(original, remedied)
```

## Arguments

| | |
|---|---|
| `original` | The original `lm` or `glm` model. |
| `remedied` | The model fitted after remediation. |

## Details

A weighted fit is shown through its Pearson residuals $\sqrt{w_i}\, e_i$, so the plot shows whether the weighting flattened the spread. Its raw residuals would look as heteroscedastic as the original ones however good the weights were.

Each panel has its own axes. Weighted residuals and the residuals of a transformed response are not in the units of the original ones, and on a common axis one of the two panels could be squeezed flat. The panels are there to compare the shape of the two clouds, not their size.

## Value

A `ggplot` object with one panel per model. Its data have the columns `fitted`, `resid`, `model` (`"original"` or `"remedied"`) and `panel`.

## Earlier versions

Up to 0.12.0 the two sets of residuals were overlaid in one panel and told apart by colour.

## Examples

```
data(mtcars)
m1 <- lm(mpg ~ wt, data = mtcars)
m2 <- fitWLS(m1)
plotBeforeAfter(m1, m2)
```

## plotBubbleVariance *Bubble plot of residual variance by covariate*

### Description

Displays residual magnitude against a predictor with bubble size proportional to `|residual|`.

### Usage

```
plotBubbleVariance(model, variable = NULL)
```

### Arguments

| | |
|---|---|
| `model` | A fitted model of class `lm` or `glm`. |
| `variable` | Optional name of a covariate from the model to plot against. |

### Value

A `ggplot` object.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
plotBubbleVariance(m, "wt")
```

## plotDiagnosticSuite *Generate a suite of diagnostic plots*

### Description

This convenience wrapper returns residual-vs-fitted and spread-level plots to help visually assess heteroscedastic patterns.

### Usage

```
plotDiagnosticSuite(model, enhanced = FALSE)
```

### Arguments

| | |
|---|---|
| `model` | A fitted model of class `lm`. |
| `enhanced` | Logical, passed to `plotResidualsFitted()`. `TRUE` returns what the deprecated `plotDiagnosticSuiteEnhanced()` returned. |

### Value

A list with elements `residuals_fitted` and `spread_level`, each a `ggplot` object.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
plots <- plotDiagnosticSuite(m)
plots$residuals_fitted
```

---

## plotDiagnosticSuiteEnhanced *Enhanced diagnostic plot suite (deprecated)*

---

### Description

Deprecated since 0.14.0: use `plotDiagnosticSuite(model, enhanced = TRUE)`, which returns the same list. This name still works and warns.

### Usage

```
plotDiagnosticSuiteEnhanced(model)
```

### Arguments

| | |
|---|---|
| `model` | A fitted model of class `lm`. |

### Value

A list of ggplot objects.

---

## plotResidualDensity *Density plot of residuals*

---

### Description

Shows the distribution of residuals with a kernel density estimate.

### Usage

```
plotResidualDensity(model)
```

### Arguments

| | |
|---|---|
| `model` | A fitted model of class `lm` or `glm`. |

### Value

A ggplot object.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
plotResidualDensity(m)
```

---

## plotResidualQQ *QQ plot of residuals*

---

### Description

Visualises departure from normality using a QQ plot.

### Usage

```
plotResidualQQ(model)
```

### Arguments

| | |
|---|---|
| model | A fitted model of class lm or glm. |

### Value

A ggplot object.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
plotResidualQQ(m)
```

---

## plotResidualsFitted *Plot residuals vs fitted values*

---

### Description

Generates a simple scatter plot of residuals against fitted values from a linear model. A horizontal reference line at zero is added.

### Usage

```
plotResidualsFitted(model, enhanced = FALSE)
```

### Arguments

| | |
|---|---|
| model | A fitted model of class lm. |
| enhanced | Logical. FALSE, the default, draws the simple plot; TRUE the plot with influential observations marked. |

### Details

For a weighted fit the Pearson residuals $\sqrt{w_i}\, e_i$ are plotted, as in plot.lm(): those are the residuals that have constant variance when the weights are right. The other residual plots in the package do the same.

With enhanced = TRUE the plot marks influential observations (Cook's distance above $4/n$) in red, sizes the points by the absolute residual and adds the standard-error band of the LOESS smooth. This is what the deprecated plotResidualsFittedEnhanced() drew.

### Value

A ggplot object.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
plotResidualsFitted(m)
plotResidualsFitted(m, enhanced = TRUE)
```

---

## plotResidualsFittedEnhanced

*Enhanced residuals vs fitted plot (deprecated)*

---

### Description

Deprecated since 0.14.0: use plotResidualsFitted(model, enhanced = TRUE), which draws the same plot. This name still works and warns.

### Usage

```
plotResidualsFittedEnhanced(model)
```

### Arguments

| | |
|---|---|
| model | A fitted model of class lm. |

### Value

A ggplot object.

## plotSpreadLevel *Spread-Level plot for variance diagnostics*

### Description

Plots the square root of the absolute residuals against fitted values. A lowess smooth is added to highlight trends.

### Usage

```
plotSpreadLevel(model)
```

### Arguments

| | |
|---|---|
| model | A fitted model of class lm. |

### Value

A ggplot object.

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
plotSpreadLevel(m)
```

## print.htest_enhanced *Print method for enhanced White test results*

### Description

Print method for enhanced White test results

### Usage

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

### Arguments

| | |
|---|---|
| x | An object of class htest_enhanced, as returned by performWhiteTestEnhanced(). |
| ... | Further arguments passed to print.htest(). |

`rcalculateEffectSize` *Effect size calculations for heteroscedasticity diagnostics*

## Description

Converts chi-squared statistics produced by heteroscedasticity tests into interpretable effect sizes such as Cramer's V, the phi coefficient, or an eta-squared analogue. The helper also provides qualitative magnitude descriptors and a brief interpretation string that can be surfaced to users.

## Usage

```
rcalculateEffectSize(
  test_result,
  model,
  data,
  type = c("cramers_v", "phi", "eta_squared")
)
```

## Arguments

| | |
|---|---|
| `test_result` | Result object from a heteroscedasticity test (typically of class `htest`). |
| `model` | Fitted model supplied to the diagnostic. |
| `data` | Data frame containing the variables referenced in `model`. |
| `type` | Effect size metric to compute. Supported options are `"cramers_v"`, `"phi"`, and `"eta_squared"`. |

## Value

A named list with elements `effect_size`, `magnitude`, `practical_significance`, `interpretation`, and `type`.

## Deprecated

Deprecated since 0.14.0: an internal helper of the package, which leaves the exports in 1.0.0. It still works and warns.

## Examples

```
data(mtcars)
model <- lm(mpg ~ wt + cyl, data = mtcars)
result <- performWhiteTest(model, mtcars)
rcalculateEffectSize(result, model, mtcars)
```

## registerDiagnostic *Register a custom diagnostic*

### Description

Adds a function to the diagnostics registry so it can be called by `runHeteroTests`.

### Usage

```
registerDiagnostic(name, fun)
```

### Arguments

| | |
|---|---|
| `name` | Name of the diagnostic. |
| `fun` | Function taking `model` and `data` arguments. |

### Details

A diagnostic registered this way is never cached by `cachedTest` or `runHeteroTests`, nor is a test of the package registered again under its name: whether it draws random numbers, or reads anything besides the model and the data, cannot be known.

### Value

Invisibly returns `NULL`.

### See Also

`listDiagnostics` for the names already registered.

### Examples

```
custom <- function(model, data) list(statistic = 0)
registerDiagnostic("custom", custom)
```

## registerPlot *Register a diagnostic plot*

### Description

Register a diagnostic plot

### Usage

```
registerPlot(name, fun)
```

### Arguments

| | |
|---|---|
| name | Name of the plot |
| fun | Function taking a model and returning a ggplot object |

### Value

Invisibly returns NULL.

### Examples

```
registerPlot("custom_plot", function(model) ggplot2::ggplot())
```

---

## restimate_test_power *Estimate test power from observed effect size*

---

### Description

Uses the observed chi-squared statistic and effect size to approximate the achieved power of a heteroscedasticity test and to suggest an increased sample size required to attain the desired power threshold.

### Usage

```
restimate_test_power(
  test_result,
  model,
  data,
  alpha = 0.05,
  target_power = 0.8,
  max_multiplier = 5
)
```

### Arguments

| | |
|---|---|
| test_result | An object returned by perform*Test() functions with statistic and parameter components. |
| model | Fitted model used in the diagnostic. |
| data | Data frame supplied to the diagnostic. |
| alpha | Significance level used in the test. Defaults to 0.05. |
| target_power | Desired minimum power for the recommendation. |
| max_multiplier | Upper bound multiplier relative to the observed sample size when searching for the recommended sample size. |

### Value

A list with elements power, recommended_n, and details containing auxiliary information (degrees of freedom, non-centrality parameter, and a power curve over a grid of effect sizes).

### Deprecated

Deprecated since 0.14.0: an internal helper of the package, which leaves the exports in 1.0.0. It still works and warns.

---

`robust_implementations` *Robust heteroscedasticity diagnostics*

---

### Description

Provides enhanced implementations of core heteroscedasticity diagnostics with bootstrap support, effect size calculations, and statistical power summaries. The module also exposes orchestration helpers for running the enhanced diagnostics in batch and utilities for validating results against reference implementations from other packages.

---

`runAdvancedDiagnostics` *Run enhanced heteroscedasticity diagnostics*

---

### Description

Executes one or more robust diagnostics, with an automatic bootstrap for small samples. The Breusch-Pagan entry is Koenker's studentized statistic, `performBPTestRobust(studentize = TRUE)`.

### Usage

```
runAdvancedDiagnostics(
  model,
  data,
  tests = "all",
  auto_enhance = TRUE,
  bootstrap_B = 999,
  parallel = FALSE,
  ci_level = 0.95
)

rrunAdvancedDiagnostics(
  model,
  data,
  tests = "all",
  auto_enhance = TRUE,
  bootstrap_B = 999,
  parallel = FALSE,
  ci_level = 0.95
)
```

### Arguments

| | |
|---|---|
| model | A fitted stats::lm object representing the mean specification to be diagnosed. |
| data | A base::data.frame (or object coercible to one) containing the variables referenced by model. It must include the observations used to fit model and will be checked for missing values. |
| tests | Character vector of tests to run. Use "all" (default) to run the White and Breusch-Pagan diagnostics, or supply a subset such as c("white", "bp"): "bp" is another name for "breusch_pagan", and the result is named breusch_pagan either way. Before 0.16.0 "bp" was refused ("Unsupported test requested: bp"). |
| auto_enhance | Logical, enable the automatic bootstrap for small samples (n < 50). It does nothing else: the Breusch-Pagan entry is the studentized statistic either way. Before 0.16.0 auto_enhance = FALSE also turned the studentization off and ran the classical Breusch-Pagan test, which assumes normal errors; performBPTestRobust() with studentize = FALSE still runs it. |
| bootstrap_B | Number of bootstrap replications used when automatic enhancement triggers bootstrap. Defaults to 999, the default of every resampling test in the package since 0.15.0; it was 500 before. On a fit returned by fitWLS() the bootstrap of performWhiteTestRobust() and performBPTestRobust() estimates the weights again in each replicate, the default of bootstrapTestStatistic() since 0.15.0, and there is no argument to hold them fixed here. |
| parallel | Logical, allow parallel bootstrap evaluation when the parallel package is available. A seeded call gives the same result each time; see *Parallel replicates* in bootstrapTestStatistic(). |
| ci_level | Confidence level for reported intervals. |

### Value

A list with the executed results and metadata describing the enhancements that were applied.

### Deprecated name

rrunAdvancedDiagnostics() is deprecated since 0.15.0: use runAdvancedDiagnostics(), the same function under the package's naming scheme. The old name still works, returns what it returned and warns; it leaves the exports in 1.0.0.

---

## runDiagnosticPlots *Run registered diagnostic plots*

---

### Description

Run registered diagnostic plots

### Usage

```
runDiagnosticPlots(
  model,
 plots = c("residuals_fitted", "spread_level", "density", "qq", "bubble_variance")
)
```

### Arguments

| | |
|---|---|
| model | A fitted lm or glm object |
| plots | Character vector of plot names to generate |

### Value

Named list of ggplot objects

### Examples

```
runDiagnosticPlots(lm(mpg ~ wt, mtcars))
```

---

## runDiagnostics *Run a suite of model diagnostics*

---

### Description

Combines heteroscedasticity checks with tests for collinearity, nonlinearity and influential observations.

### Usage

```
runDiagnostics(
  model,
  data = NULL,
  tests = c("white", "koenker"),
  power = 2:3,
  use_cache = TRUE,
  chunk_threshold_mb = Inf,
  chunk_size = 10000,
  progress = interactive()
)
```

### Arguments

| | |
|---|---|
| model | A fitted lm, or a glm with the Gaussian family and the identity link, or a formula. The tests refuse a glm of any other family or link since 0.15.0. |
| data | Optional data frame if model is a formula. |

| | |
|---|---|
| tests | Heteroscedasticity tests to run. Defaults to the White and Koenker tests, the default of runHeteroTests; it was the White and the classical Breusch-Pagan tests up to 0.14.0. |
| power | Powers for performRESETTest. |
| use_cache | Logical flag forwarded to runHeteroTests controlling whether cached diagnostic results may be reused. |
| chunk_threshold_mb | Size of the data in megabytes above which runHeteroTests runs the streaming forms of the White, Breusch-Pagan and Koenker tests. The default, Inf, never streams; give a finite value to ask for them. Up to 0.15.0 the default was 100. |
| chunk_size | Integer chunk size supplied to runHeteroTests when the streaming tests run. |
| progress | Logical flag controlling whether textual progress indicators are displayed during long-running operations. |

## Value

A list with the results of all diagnostics.

## Examples

```
data(mtcars)
runDiagnostics(mpg ~ wt + qsec, mtcars)
```

---

# runHeteroTests *Run multiple heteroscedasticity diagnostics*

---

## Description

Convenience wrapper that executes several heteroscedasticity tests on a fitted linear model.

## Usage

```
runHeteroTests(
  model,
  data = NULL,
  tests = c("white", "koenker"),
  use_cache = TRUE,
  chunk_threshold_mb = Inf,
  chunk_size = 10000,
  progress = interactive()
)
```

## Arguments

| | |
|---|---|
| model | a fitted lm, or a glm with the Gaussian family and the identity link, a model formula, a tidymodels workflow or a parsnip model fit. The tests refuse a glm of any other family or link since 0.15.0, the deprecated registry names "ncv" and "student_bp" and the substitutes of the fallback (Details) included: the call stops with the error of the first test asked for, and with grouped data each test is returned as failed. A workflow, or a parsnip fit, is tested on the fit of its engine, which names its response ..y and its predictors as the workflow's preprocessing made them: the data the workflow was fitted to are mapped to those variables by that preprocessing (hardhat::forge()), the outcome as the preprocessing transforms it among them, or the outcome of a fit of parsnip::fit_xy() to ..y, and once their rows are matched to the observations of the fit, every column of its model frame compared, its model frame is tested, as without data. Before 0.17.0 they were refused. Data in another order than the workflow was fitted in are refused, and so are data that cannot be mapped, such as data without a column the preprocessing needs, grouped data, and the data of a workflow whose recipe skips on new data a step that changed the outcome or a predictor at the fit, such as step_log(y, skip = TRUE), which the message names. The case weights of a workflow are compared with those of the fit, on the rows its preprocessing keeps; the data of a parsnip fit of fit_xy() with case weights, whose column the fit does not record, are refused. Each result keeps what generics::augment() needs to map the data in the same way, the blueprint of the preprocessing, the number of rows it kept and the column of the case weights, or the outcome of a parsnip fit, as attr(, "tidymodels"), the blueprint as the workflow keeps it: a recipe step keeps the environment it was written in, so that a saved result carries what that environment holds, as saving the workflow does. The results are otherwise those of the fit of the engine. |
| data | optional data frame used to fit model. If omitted, model.frame(model) is used, which serves only the tests that find what they need there (see their data argument). With data grouped by dplyr::group_by() the model is refitted within each group by evaluating its own call on that group, so every variable it uses, including weights, offset and subset, must be a column of data. A fit returned by fitWLS() is refused with grouped data since 0.17.0: its call does not hold the weights it estimated, so each group was tested as the unweighted fit of its rows; fit fitWLS() within each group, or pass the data ungrouped. So is a MASS::rlm() fit, which every test refuses. A test that refuses the model of a group as the model context refuses a model, such as a multivariate lm or a penalised mgcv::gam() fit given a test that needs least squares, stops the grouped run with that refusal and its class since 0.17.0, as the run on ungrouped data does; before, the group recorded it as a failed test, whose statistic was NA. The failures of a test for another reason are recorded in its group, as before. |
| tests | character vector of test names to run: any of the names listDiagnostics() returns, which include those added with registerDiagnostic. On a fit returned by fitWLS() the resampling tests "wild_bootstrap" and "white_bootstrap" estimate the weights again in each replicate, as performWildBootstrapTest() and performWhiteTestBootstrap() do by default since 0.15.0; to hold them fixed, call those functions with reestimate_weights = FALSE. |

use_cache — logical: reuse the results of an earlier call on the same inputs? Requires the **digest** package and defaults to TRUE. Only deterministic tests are cached, under a key that holds the test and the parts of the model and the data it reads (see cachedTest, which also says what the key leaves out): the resampling tests "wild_bootstrap", "white_bootstrap", "rank_permutation" and "spatial_hetero", and diagnostics added with registerDiagnostic, run on every call, and a call that runs one, or falls back on one, is not stored as a whole. A seeded call returns what it returns with use_cache = FALSE, identical(), and leaves the random number generator where it leaves it. Each test is kept in the cache of cachedTest, and each call whose tests are all deterministic in a second cache (clearAnalysisCache); each holds at most getOption("heteroTests.cache_size", 100) results and drops the one used least recently, and options(heteroTests.cache_size = 0) turns both off. The checks the tests make first have a cache of their own, bounded by getOption("heteroTests.validation_cache_size", 1000). Before 0.16.0 the resampling tests were cached too, and two fits with the same coefficients and residuals shared their results.

chunk_threshold_mb — size of data in megabytes above which "white", "breusch_pagan" and "koenker" are run by their streaming forms. The default, Inf, never streams: give a finite value to ask for them, such as 100, or 0 for data of any size. The result of a streaming test has attr(, "extras")$streaming set to TRUE. Up to 0.15.0 the default was 100. A warning about the size of the data is given above half the threshold, or above 50 MB with the default.

chunk_size — integer number of observations processed per chunk when the streaming tests run (chunk_threshold_mb).

progress — logical flag controlling whether textual progress bars are displayed when running multiple diagnostics.

## Details

By default runs performWhiteTest and performKoenkerTest, the studentized Breusch-Pagan test, but additional tests can be requested. Custom diagnostics registered via registerDiagnostic are also available.

Up to 0.14.0 the default was c("white", "breusch_pagan"), whose second test is the classical Breusch-Pagan statistic of performBPTest. That statistic divides by $2\hat{\sigma}^4$, the variance of the squared errors only when they are normal. In the full sweep of inst/validation/, with 150 observations and 400 replications, it rejected a true null hypothesis 28.75% of the time at the 5% level when the errors had a $t_5$ distribution, against 4.75% for Koenker's statistic. The classical test is still run by name, tests = "breusch_pagan".

When a test stops with an error, the substitute registered for it is run, if one is and it succeeds, and its result is returned under the name of the test asked for: "white" falls back on White's test without cross products, the classical Breusch-Pagan, Koenker and Cook-Weisberg tests, "koenker" on the Cook-Weisberg test. The method of the result names the test that ran, attr(result, "diagnostic") its registry name and attr(result, "extras")$status is "fallback". A weighted fit whose weights are a column of the data, called without data, is such a case: White's and Koenker's tests stop, and both names hold the Cook-Weisberg test. Pass data for such a fit.

The runner runs the exact tests unless chunk_threshold_mb asks for the streaming ones, performWhiteTestStreaming(), performBPTestStreaming() and performKoenkerTestStreaming(). Up to 0.15.0 it switched to

them by itself on data above 100 MB, and performWhiteTest(), performBPTest() and performKoenkerTest() did so on their own. Streaming saves memory mainly in White's test, whose auxiliary design is several times wider than the model matrix; the auxiliary design of the Breusch-Pagan and Koenker tests is the model matrix itself. And until 0.17.0 the streaming White test stopped on any model with a factor or a poly() term, whose squares are collinear with the other terms of its auxiliary regression, so that above 100 MB such a model got a fallback under the name "white" where smaller data got White's test; since 0.17.0 it drops those terms, as performWhiteTest() does.

## Value

A hetero_test_suite: a named list of htest objects, one for each test asked for, which prints as a table of the tests, and whose summary() is laid out as that of an lm (see summary.hetero_test_suite). With grouped data, a hetero_grouped_suite holding one such suite for each group. Since 0.16.0 a suite whose model the runner fitted itself, from a formula, to the data of a survey design or within a group, says so in attr(, "model_origin") ("formula", "survey" or "group"; see hetero_test_suite).

A result whose data.name is "model" reports the formula of the model instead, as Koenker's test and the others do. Up to 0.15.0 five tests reported "model" here, the name of the argument of the function the registry calls them with: White's test, its bootstrap and its form without cross products, the studentized Breusch-Pagan statistic and Szroeter's test. Any other data.name, such as that of a diagnostic added with registerDiagnostic, is kept. Called directly, a test still names the object it is given: performWhiteTest(m, mtcars)$data.name is "m".

## References

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4), 817–838. Breusch, T. S., & Pagan, A. R. (1979). A Simple Test for Heteroscedasticity and Random Coefficient Variation. *Econometrica*, 47(5), 1287–1294. Koenker, R. (1981). A note on studentizing a test for heteroscedasticity. *Journal of Econometrics*, 17(1), 107–112. doi:10.1016/03044076(81)900622

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
# Default: White and Koenker
runHeteroTests(m, mtcars)
# Laid out as summary.lm() is, with the significance codes
summary(runHeteroTests(m, mtcars))
# Specify additional diagnostics, or the classical Breusch-Pagan test
runHeteroTests(m, mtcars, tests = c("white", "koenker", "cook_weisberg"))
runHeteroTests(m, mtcars, tests = "breusch_pagan")
custom <- function(model, data) list(stat = 1)
registerDiagnostic("custom", custom)
runHeteroTests(m, mtcars, tests = c("white", "custom"))
```

`runHeteroTestsParallel` *Parallel test execution helper*

## Description

Executes a set of registered diagnostics in parallel when multiple CPU cores are available, falling back to sequential execution otherwise.

## Usage

```
runHeteroTestsParallel(model, data, tests, n_cores = NULL)
```

## Arguments

| | |
|---|---|
| `model` | Fitted model object passed to each diagnostic. Must be compatible with the internal `.test_factory` registry. |
| `data` | Data frame used to fit `model` and supplied to each diagnostic. |
| `tests` | Character vector naming the diagnostics to execute. |
| `n_cores` | Optional positive integer giving the number of worker processes to spawn. Defaults to `parallel::detectCores() - 1`, with a lower bound of 1. |

## Details

When `n_cores` exceeds one the diagnostics run on a socket cluster of that many workers (`parallel::makeCluster()`), started for the call, each of which loads the installed heteroTests. The model and the data travel to the workers with each task. With `n_cores = 1`, or when the default comes to one worker, the diagnostics run in the session.

Test `i` draws its random numbers from stream `i` of a L'Ecuyer-CMRG generator seeded by one draw from the session's generator, on the workers and in the session alike. A seeded call that runs a resampling test, such as `"wild_bootstrap"`, therefore returns the same value each time and whatever `n_cores`, `n_cores = 1` included. It is not the value that test returns when called directly from the same seed, which draws from the session's generator; both are bootstraps of the same data.

## Value

Named list of test results in the order provided by `tests`.

## See Also

`cachedTest()` for memoised execution of individual diagnostics.

## runMultivariateTests *Run multivariate heteroscedasticity tests*

### Description

Convenience wrapper for multivariate tests such as Box's M.

### Usage

```
runMultivariateTests(data, group, tests = c("box_m"))
```

### Arguments

| | |
|---|---|
| data | Numeric data frame or matrix with multiple variables. |
| group | Factor defining groups. |
| tests | Character vector of test names. |

### Value

A named list of htest objects.

### Examples

```
runMultivariateTests(iris[,1:4], iris$Species)
```

## runPanelTests *Run panel-data heteroscedasticity tests*

### Description

Breusch-Pagan LM and Pesaran CD tests for panel models.

### Usage

```
runPanelTests(model, data, id, time = NULL,
          tests = c("bp_random", "pesaran"))
```

### Arguments

| | |
|---|---|
| model | A fitted lm object. |
| data | Data frame used to fit model. The two tests read id and time from the rows of data that hold the observations of the fit: since 0.16.0, data that are given are matched to the observations of the fit, by row name in any order, or by the row numbers or in the order of the fit for rows that only number themselves, as described under data in performPesaranTest(). |
| id | Column identifying individuals. |
| time | Column identifying time periods for the Pesaran test, whose pairs of id and time must each hold one observation of the fit. |
| tests | Character vector of test names. |

### Value

A named list of htest objects.

### See Also

runTimeSeriesTests

### Examples

```
df <- data.frame(id = rep(1:3, each = 4), time = rep(1:4, 3),
                 x = runif(12), y = rnorm(12))
m <- lm(y ~ x, data = df)
runPanelTests(m, df, id = "id", time = "time")
```

---

## runSurveyHeteroTests *Run heteroscedasticity diagnostics on the data of a survey design*

---

### Description

A convenience wrapper that takes the data out of a survey::svydesign() object, fits the model by ordinary least squares and runs runHeteroTests. **It does not use the sampling weights, strata or clusters of the design**, and warns when the design has unequal weights or clusters.

### Usage

```
runSurveyHeteroTests(formula, design, tests = c("white", "koenker"), ...)
```

### Arguments

| | |
|---|---|
| formula | Model formula specifying the mean structure. |
| design | A survey::svydesign() object. Only its data are used. |
| tests | Diagnostic names passed on to runHeteroTests. Defaults to the White and Koenker tests, the default of runHeteroTests; it was the White and the classical Breusch-Pagan tests up to 0.14.0. |
| ... | Additional arguments forwarded to runHeteroTests. |

## Details

The tests are the ones runHeteroTests runs on any lm: they treat the observations as an independent sample with equal weights. They hold their level on survey data when the design can be ignored given the regressors, that is when the chance of being sampled depends only on variables in the model, and the sample has no clusters. When selection depends on the response, or observations are clustered, they reject a model with constant error variance far more often than their nominal level.

The package has no design-based test of constant variance. Sampling weights are not precision weights: they say how many population units an observation stands for and nothing about its error variance. Passing them to lm(weights = ), or testing a survey::svyglm() fit, makes the tests examine residuals multiplied by the square roots of the weights, and a homoscedastic model is then rejected whenever the weights vary with the regressors. svyglm fits are refused for that reason.

inst/validation/survey-designs-size.R measures all of this, together with two design-based tests that were tried and not adopted because they reject too often at the sample sizes examined.

## Value

An object of class hetero_test_suite. Since 0.16.0 it carries attr(, "model_origin") = "survey": the model is fitted here, so summary() shows its formula rather than its call (see hetero_test_suite).

## Earlier versions

Up to 0.12.0 this page described a survey-weighted fit. The weights did not reach the fit, which was by ordinary least squares, as it is now, so the reported values are unchanged.

## See Also

runHeteroTests

## Examples

```
if (requireNamespace("survey", quietly = TRUE)) {
  data(api, package = "survey")
  design <- survey::svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat)
  # Warns: the weights of this design are unequal and are not used.
  res <- runSurveyHeteroTests(api00 ~ api99 + ell, design)
  generics::tidy(res)
}
```

---

# runTimeSeriesTests *Run time-series heteroscedasticity tests*

---

## Description

Convenience wrapper for Engle's ARCH LM and McLeod-Li tests.

### Usage

```
runTimeSeriesTests(model, lags = 1, tests = c("arch_lm", "mcleod_li"))
```

### Arguments

| | |
|---|---|
| model | A fitted lm object. |
| lags | Number of lags for both tests. |
| tests | Character vector of test names. |

### Value

A named list of htest objects.

### See Also

runPanelTests

### Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, mtcars)
runTimeSeriesTests(m, lags = 2)
```

---

## run_benchmark_suite *Benchmark* **heteroTests** *against reference implementations*

---

### Description

run_benchmark_suite() evaluates a selection of heteroscedasticity diagnostics across varying sample sizes while measuring runtime, memory allocation, and statistical agreement with the **lmtest** and **car** packages. The companion generate_benchmark_report() helper aggregates the raw output into tidy summary tables and recommendations.

### Usage

```
run_benchmark_suite(
  sample_sizes = c(100L, 500L, 1000L, 10000L, 100000L, 1000000L),
  tests = NULL,
  replicates = 3L,
  hetero_patterns = c("none", "linear", "group"),
  hetero_strength = 1,
  baseline_packages = c("lmtest", "car"),
  seed = 123L,
  n_predictors = 4L,
  profile_memory = TRUE,
  progress = interactive()
)

generate_benchmark_report(benchmark_results, accuracy_tolerance = 1e-4)
```

## Arguments

| | |
|---|---|
| `sample_sizes` | Integer vector of observation counts to benchmark. Values may range from a few hundred observations to one million. |
| `tests` | Optional character vector restricting the benchmark to specific diagnostics (currently `"breusch_pagan"`, `"koenker"`, and `"ncv"`). |
| `replicates` | Scalar integer applied to all sample sizes or an integer vector matching `sample_sizes` to control the number of repetitions per configuration. |
| `hetero_patterns` | Character vector describing the heteroscedasticity patterns used when generating synthetic data. Supported values are `"none"`, `"linear"`, `"group"`, and `"exponential"`; the vector is recycled across replications. |
| `hetero_strength` | Positive numeric multiplier controlling the severity of variance heterogeneity. |
| `baseline_packages` | Character vector of external packages used for comparisons. Missing dependencies are skipped and reported in the metadata. |
| `seed` | Integer seed passed to the data generator for reproducibility. |
| `n_predictors` | Number of regressors (excluding the intercept) used when simulating benchmark datasets. |
| `profile_memory` | Logical; when `TRUE` and the **bench** package is installed, the suite records peak memory allocations in megabytes. |
| `progress` | Logical indicating whether to display a textual progress bar during execution. |
| `benchmark_results` | List returned by `run_benchmark_suite()`. |
| `accuracy_tolerance` | Non-negative numeric tolerance for declaring statistical agreement with reference implementations (based on absolute p-value differences). |

## Details

Datasets are generated from a linear regression model with a configurable number of predictors and variance patterns (none, linear, group, or exponential). Each diagnostic from **heteroTests** is executed alongside equivalent tests from **lmtest** and **car** when available. Memory profiling is enabled via **bench** when installed; otherwise only runtimes are collected. The output is suitable for regression testing (to detect performance regressions) and for building artefacts such as CSV benchmark reports.

## Value

`run_benchmark_suite()` returns a list with four elements:

**performance** Data frame containing runtime, memory, and inference metrics for every test, sample size, and replicate.

**accuracy** Data frame recording absolute differences in p-values and test statistics relative to each baseline implementation.

**summary** List of aggregated medians for quick inspection of speed, memory, and accuracy.

**metadata** Metadata describing the benchmark configuration (sample sizes, baselines, seed, and availability of optional dependencies).

`generate_benchmark_report()` converts the raw results into aggregated speed, memory, accuracy, recommendation, and scalability tables alongside the original metadata.

## Examples

```
if (requireNamespace("lmtest", quietly = TRUE) &&
    requireNamespace("car", quietly = TRUE)) {
  results <- run_benchmark_suite(
    sample_sizes = c(100L, 250L),
    replicates = 1L,
    profile_memory = FALSE,
    progress = FALSE
  )
  report <- generate_benchmark_report(results)
  head(report$recommendations)
}
```

---

`rvalidate_against_reference` *Validate diagnostics against reference implementations*

---

## Description

Compares the package's test results with established implementations from other packages (currently `lmtest` and `car`). Differences exceeding the supplied tolerance are flagged.

## Usage

```
rvalidate_against_reference(test_name, model, data, tolerance = 1e-06)
```

## Arguments

| | |
|---|---|
| `test_name` | Character scalar naming the diagnostic: "breusch_pagan", "studentized_bp", or "white". |
| `model` | A fitted stats::lm object representing the mean specification to be diagnosed. |
| `data` | A base::data.frame (or object coercible to one) containing the variables referenced by `model`. It must include the observations used to fit `model` and will be checked for missing values. |
| `tolerance` | Acceptable absolute difference between statistics and p-values when comparing to the reference implementation. |

## Value

A list describing the comparison, including a status field with values `"ok"`, `"mismatch"`, or `"skipped"` when the reference package is unavailable.

## Deprecated

Deprecated since 0.14.0: an internal helper of the package, which leaves the exports in 1.0.0. It still works and warns.

---

`scale_colour_hetero_diagnostic` *Colour scale for heteroscedasticity diagnostics*

---

## Description

Provides a discrete palette used across diagnostic comparisons.

## Usage

```
scale_colour_hetero_diagnostic(...)

scale_fill_hetero_diagnostic(...)
```

## Arguments

| | |
|---|---|
| `...` | Arguments passed to `ggplot2::scale_colour_manual()`. |

## Value

A ggplot2 scale.

## Examples

```
ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg, colour = factor(cyl))) +
  ggplot2::geom_point() +
  scale_colour_hetero_diagnostic()
```

simulate_hetero *Simulate heteroscedastic data*

## Description

Utilities to generate data with non-constant variance patterns. simulate_hetero simulates a linear regression model with heteroscedastic errors using a user-supplied variance function. The helper functions prefixed by sigma_ implement common variance patterns and can be passed to sigma_func. simulate_arch1 creates a simple ARCH(1) time series.

## Usage

```
simulate_hetero(n, beta0, beta1, sigma_func, seed = NULL)

simulate_arch1(n, mu = 0, alpha0 = 0.5, alpha1 = 0.3, seed = NULL)

sigma_linear(x)

sigma_exponential(x)

sigma_group(x)

sigma_piecewise(x)

sigma_poly(x, a = 0.5, b = 0.1, c = 0.02)

sigma_sin(x, A = 1, B = 0.5, omega = 2 * pi/10, phi = 0)

sigma_multiplicative(x, mu_func, p = 1)

sigma_spatial(coords, gamma0 = 0.5, gamma1 = 0.1)

sigma_logistic(x, L = 2, k = 1, x0 = 5)

sigma_inverse(x, a = 1, b = 1)

sigma_power(x, a = 0.5, p = 0.5)

sigma_step(x, thr = 5, low = 0.5, high = 2)

sigma_u_shape(x, a = 0.1, b = 0.05, center = 5)

sigma_exp_decay(x, a = 2, b = 0.2)

sigma_gaussian_peak(x, base = 0.5, height = 1, mu = 5, sd = 1)

sigma_piecewise_linear(x, thr = 5, slope1 = 0.1, slope2 = 0.3)
```

## Arguments

| | |
|---|---|
| n | Number of observations for simulate_hetero, or the length of the series for simulate_arch1. |
| beta0 | Intercept of the regression. |
| beta1 | Slope of the regression. |
| sigma_func | Function returning the standard deviation for each x. |
| seed | Optional integer seed for reproducibility. |
| mu | Mean of the ARCH(1) process for simulate_arch1, and the peak location for sigma_gaussian_peak. |
| alpha0 | Constant term in the ARCH variance equation. |
| alpha1 | ARCH coefficient (must be in [0,1)). |
| x | Numeric vector used by the sigma_* functions. |
| coords | Matrix or data frame with columns x and y for sigma_spatial. |
| mu_func | Function returning mean values for sigma_multiplicative. |
| p | Exponent for sigma_multiplicative or sigma_power. |
| a, b, c, A, B, omega, phi, L, k, x0, thr, low, high, center, base, height, sd, slope1, slope2, gamma0, gamma1 | Additional numeric parameters controlling the shape of the variance patterns. |

## Value

For simulate_hetero a data frame with columns x and y. For simulate_arch1 a data frame with columns time, y and sigma. The sigma_* helpers return numeric vectors of standard deviations.

## Naming

The simulation functions form one family and are named in snake_case on purpose: simulate_hetero(), simulate_arch1(), simulate_size() and simulate_power(), and the sigma_*() variance functions. The rest of the package uses camelCase. The family keeps these names after 1.0.0.

## Examples

```
set.seed(1)
hetero <- simulate_hetero(200, 1, 2, sigma_linear)
head(hetero)

# Other variance patterns
simulate_hetero(100, 0, 1, sigma_logistic)
simulate_arch1(50)
```

`simulation_framework` *Advanced simulation framework for test validation*

## Description

Provides tools for validating heteroscedasticity tests under controlled scenarios and estimating statistical power.

`simulate_type_I_errors()` and `simulate_power_analysis()` are deprecated since 0.14.0: use `simulate_size()` and `simulate_power()`, which return the same results and belong to the documented snake_case family of `simulate_*()` and `sigma_*()` functions. The old names still work and warn.

## Usage

```
simulate_size(
  test_function,
  n_sims = 1000,
  alpha = 0.05,
  n_obs = 100,
  seed = 123
)

simulate_power(
  test_function,
  sigma_functions = list(sigma_linear),
  effect_sizes = c(0.1, 0.2, 0.5, 1),
  n_sims = 500,
  n_obs = 100,
  alpha = 0.05
)

simulate_type_I_errors(
  test_function,
  n_sims = 1000,
  alpha = 0.05,
  n_obs = 100,
  seed = 123
)

simulate_power_analysis(
  test_function,
  sigma_functions = list(sigma_linear),
  effect_sizes = c(0.1, 0.2, 0.5, 1),
  n_sims = 500,
  n_obs = 100,
  alpha = 0.05
)
```

### Arguments

| | |
|---|---|
| test_function | Function taking a model and data and returning an object with a numeric p.value component. |
| n_sims | Number of simulation replications. |
| alpha | Significance level for Type I error or power calculations. |
| n_obs | Number of observations for generated datasets. |
| seed | Optional integer seed for reproducibility. |
| sigma_functions | List of variance functions generating heteroscedastic patterns. |
| effect_sizes | Numeric vector of effect size multipliers. |

### Value

A list summarising Type I error rate or a data frame of power estimates.

---

## std_error *Generate standardized error message (deprecated)*

---

### Description

Deprecated since 0.14.0: an internal helper of the package, which leaves the exports in 1.0.0. It still works and warns.

### Usage

```
std_error(type, ...)
```

### Arguments

| | |
|---|---|
| type | Error message type. |
| ... | Named arguments to replace in the template. |

### Value

Stops execution with a formatted message.

## std_warning *Generate standardized warning message (deprecated)*

### Description

Deprecated since 0.14.0: an internal helper of the package, which leaves the exports in 1.0.0. It still works and warns.

### Usage

```
std_warning(type, ...)
```

### Arguments

| | |
|---|---|
| type | Warning message type. |
| ... | Named arguments for the template. |

### Value

Issues a warning with a formatted message.

## suggestRemediation *Suggest remediation actions for heteroscedasticity*

### Description

Given diagnostic test results from `runHeteroTests()`, this helper provides a basic summary of recommended follow-up steps. It evaluates the number of significant tests and proposes variance stabilising transformations or modelling approaches.

### Usage

```
suggestRemediation(diagnostic_results)

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

### Arguments

| | |
|---|---|
| diagnostic_results | Named list of `htest` objects as returned by `runHeteroTests()`. |
| x | Object of class `remediation_suggestions`. |
| ... | Not used. |

### Details

The function counts how many diagnostic tests yield a p-value below 0.05. If none are significant it returns a brief conclusion that no action is needed. Otherwise a severity level is assigned and appropriate transformations or variance modelling approaches are suggested: a transformation when White's test rejects, and variance modelling when the Breusch-Pagan test (`"breusch_pagan"`) or Koenker's studentized form of it (`"koenker"`, in the default battery of `runHeteroTests()` since 0.15.0) rejects.

### Value

An object of class `remediation_suggestions` containing a summary of potential actions.

### Examples

```
data(mtcars)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
res <- runHeteroTests(mod, mtcars)
suggestRemediation(res)
```

---

## summary.HeteroDiagnostic

*Summarize diagnostics*

---

### Description

Summarize diagnostic test results.

### Usage

```
## S3 method for class 'HeteroDiagnostic'
summary(object, tests = c("white", "koenker"), ...)
```

### Arguments

| | |
|---|---|
| `object` | A `HeteroDiagnostic` object. |
| `tests` | Character vector naming the diagnostics to summarise. Defaults to the White and Koenker tests, the default of `runHeteroTests`; it was the White and the classical Breusch-Pagan tests up to 0.14.0. |
| `...` | Further arguments passed to the individual tests. |

### Value

Named numeric vector of test statistics.

summary.hetero_test_suite *Print and summarise the results of runHeteroTests()*

## Description

`print()` shows a `hetero_test_suite` compactly: the model and the number of observations, on one line when it fits the console, then one row per test with its statistic, degrees of freedom, p-value and significance codes, and a note for each test that did not run as asked. `summary()` returns an object of class `summary.hetero_test_suite`, which prints as the summary of an `lm` does (`summary.lm()`): the call, the number of observations, the table of tests, the legend of the significance codes and the notes; and `coef()` of it returns the table, as `coef(summary(fit))` does for an `lm`.

The `hetero_grouped_suite` of a grouped run has the same methods, with one table for all the groups, a row for each test in each group.

## Usage

```
## S3 method for class 'hetero_test_suite'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

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

## S3 method for class 'summary.hetero_test_suite'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'hetero_grouped_suite'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'hetero_grouped_suite'
```

```
summary(object, ...)

## S3 method for class 'summary.hetero_grouped_suite'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'summary.hetero_test_suite'
coef(object, ...)

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

## Arguments

| | |
|---|---|
| x | A hetero_test_suite or hetero_grouped_suite, as runHeteroTests() returns, or its summary. |
| digits | The number of significant digits to print, as in print.summary.lm(). |
| signif.stars | Logical: show the significance codes? |
| ... | Passed on to printCoefmat() by the print methods, replacing the arguments they pass (Details); unused by the summary and coef() methods. |
| object | A hetero_test_suite or hetero_grouped_suite for summary(), or its summary for coef(). |

## Details

The table is printed by printCoefmat(), so its format is that of summary.lm(): digits sets the significant digits, the p-values are formatted by format.pval(), and the significance codes and their legend are shown when signif.stars is TRUE (by default getOption("show.signif.stars")) and a p-value is below 0.1. The print methods pass printCoefmat() cs.ind = integer(), tst.ind = 1, zap.ind = integer(), P.values = TRUE, has.Pvalue = TRUE and na.print = "NA"; an argument of the same name given in ... replaces the value passed, so that print(s, na.print = "-") prints a failed test with dashes and print(s, P.values = FALSE) leaves the p-values out.

The rows are named as the suite is, by the names asked for in tests. The df column holds the degrees of freedom of the statistic: the parameter of the test named df, df1 or num df (as var.test() and anova() name it), or the first one when they have no names. The second degrees of freedom of an F statistic, named df2 or denom df, are a column of their own, df2, when a test has them. A parameter of another name, such as the number of replicates B of a resampling test or the number of observations n of Szroeter's test, is not a degree of freedom, and the notes give it.

The notes say which test did not run as asked: one whose fallback ran under its name, saying which test ran in its place and why; one that was refused or failed, with its message and, for a failure of a particular kind, that kind (attr(, "extras")$issue, such as "rank_deficiency"); one computed by a streaming implementation; a parameter that is not a degree of freedom; and a result that is not a test, as a diagnostic added with registerDiagnostic() may return. A test "refused" its input

when it stopped before computing anything, with one of the refusals of the package: of the model or of `data` by the model context (the issues `"model_context"` and `"data_rows"`), of a `glm` that is not a least-squares fit or a model of another class, of a `survey::svyglm()` fit, of zero weights, of a weighted fit by a test that refits without its weights, of an aliased coefficient, or of too few observations. These are recognised by their messages, since the runner records the message and the issue and not the class of the condition; a test that stopped otherwise "failed". `print()` gives each note on one line, cut to the width of the console; `summary()` gives it whole. Without groups, `runHeteroTests()` stops when a test fails and no fallback runs, so a failed test is returned only by a grouped run, which records the failure and goes on. Whether a result was read from a cache is not shown: a result read from it is `identical()` to the one computed (see `cachedTest()`).

Several tests of one model are not independent evidence: they are computed from the same residuals, most of them from regressions of their squares on overlapping sets of regressors. A small p-value from one and a large one from another describe the alternatives each test is built to detect; they are not two votes, and `summary()` says so.

Neither method changes the results: `generics::tidy()`, `generics::glance()`, `generics::augment()` and `ggplot2::autoplot()` read them as before.

## Value

The print methods return `x`, invisibly. `coef()` of a summary returns its `tests`.

`summary()` of a `hetero_test_suite` returns a list of class `summary.hetero_test_suite` with:

`call` The call of the model, as `summary.lm()` keeps it, or `NULL` when its call is not recorded or the model was fitted by `runHeteroTests()` itself, whose call names the runner's variables rather than yours (`stats::lm(formula = model, data = prep$data)`): the fit of a formula, the refit of a model within each group, and the fit to the data of a survey design (`runSurveyHeteroTests()`). The runner records those in the suite, as `attr(, "model_origin")` (`"formula"`, `"group"` or `"survey"`; see `hetero_test_suite`); a model you fitted is shown by its call, whatever its form.

`formula` The formula of the model, or `NULL`.

`model_class` The class of the model, such as `"lm"`.

`nobs` The number of observations, `stats::nobs()` of the model.

`na.action` The `na.action` of the model, as `summary.lm()` keeps it: the observations dropped for missing values, or `NULL`.

`weights` How the model is weighted, as text: the `weights` argument of its call (`"1/hp"`); the variance function of a `fitWLS()` fit; or, for prior weights its call does not record, their range; `NULL` for an unweighted model.

`offset` The offset of the model, as text, or `NULL`.

`family` The family and link of a `glm`, as text, or `NULL`.

`tests` A numeric matrix with one row per test, named as the suite is, and the columns `Statistic`, `df`, `df2` (only when a test has second degrees of freedom) and `p-value`, as `coef(summary(fit))` holds those of the coefficients of an `lm`; `coef(summary(s))` returns it. A failed test has `NA` in every column.

`method` A character vector, named as the suite is: the `method` of each result, the name of the test that ran.

`diagnostic` A character vector: the registry name of the test that ran, `attr(, "diagnostic")`, which differs from the name asked for when a fallback ran in its place.

`status` A character vector: `"ok"`; `"fallback"` when another test ran in its place; `"error"` when it failed; or `"other"` for a result that is not an `htest`.

`notes` A character vector of the notes, as `summary()` prints them.

`summary()` of a `hetero_grouped_suite` returns a list of class `summary.hetero_grouped_suite` with:

`formula`, `model_class` As above, for the model of the first group.

`keys` The grouping keys, one row per group, `attr(, "group_keys")` of the suite.

`nobs` An integer vector: the number of observations of each group.

`tests` A data frame with one row per test in each group: the grouping keys, `test` (the name asked for), `Statistic`, `df`, `df2` (only when a test has second degrees of freedom), `p-value` and `status`, as above; `coef(summary(g))` returns it.

`groups` A list with the `summary.hetero_test_suite` of each group.

`notes` A character vector of the notes of every group, each starting with the label of its group.

## Earlier versions

Up to 0.15.0 a suite had no print or summary method: `print()` showed the list, each test and then the attributes of the suite, the fitted model among them, and `summary()` gave the summary of a list.

## See Also

`runHeteroTests()`; `generics::tidy()` and `generics::glance()` for the results as data frames; `printCoefmat()`.

## Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
s <- runHeteroTests(m, mtcars, tests = c("white", "koenker", "cook_weisberg"))
s
summary(s)
coef(summary(s))
print(summary(s), signif.stars = FALSE)

# White's test needs 20 observations: the runner falls back on the
# Breusch-Pagan test, and the note says so.
small <- mtcars[1:18, ]
runHeteroTests(lm(mpg ~ wt, data = small), small)

if (requireNamespace("dplyr", quietly = TRUE)) {
  by_am <- dplyr::group_by(mtcars, am)
  g <- runHeteroTests(mpg ~ wt, by_am, tests = "koenker")
  print(g)
  print(summary(g)$tests)
}
```

## test.HeteroDiagnostic *Run diagnostic tests*

### Description

Run the registered diagnostics on the model.

### Usage

```
test(object, ...)
```

### Arguments

| | |
|---|---|
| object | A HeteroDiagnostic object. |
| ... | Further arguments passed to the individual tests. |

### Value

List of test results.

### Deprecated

test() is deprecated since 0.14.0: its name masks devtools::test(). Use runDiagnostics(x$model, x$data, tests), which returns what test(x, tests) returns. It still works and warns; summary() of a HeteroDiagnostic object is unchanged.

## TestFactory *Heteroscedasticity Test Factory*

### Description

Provides a registry of heteroscedasticity tests with metadata and convenience methods to run them with input validation. This is an alternative to the older environment-based registry used by runHeteroTests().

### Deprecated

TestFactory and its shared instance test_factory are deprecated since 0.14.0: they are the package's internal machinery and leave the exports in 1.0.0. They are objects rather than functions, so they cannot warn when used; this page and NEWS.md are the notice. Use registerDiagnostic() to add a test and runHeteroTests() to run one.

### Usage

The shared instance test_factory can be used to register new tests and execute them. See register and run_test methods for details.

## Methods

### Public methods:

- TestFactory$register()
- TestFactory$get_available()
- TestFactory$run_test()
- TestFactory$clone()

**Method** `register()`**:**

*Usage:*

```
TestFactory$register(name, func, metadata = list())
```

*Arguments:*

`name` Name of the test

`func` Function with arguments `model` and `data`

`metadata` Optional list with metadata fields

*Returns:* Invisibly returns the factory Get available tests

**Method** `get_available()`**:**

*Usage:*

```
TestFactory$get_available(data_type = NULL, min_n = NULL)
```

*Arguments:*

`data_type` Filter by supported data type

`min_n` Filter by minimum observations

*Returns:* Character vector of test names Run a registered test

**Method** `run_test()`**:**

*Usage:*

```
TestFactory$run_test(test_name, model, data, ...)
```

*Arguments:*

`test_name` Name of the test

`model` Fitted model

`data` Data frame used to fit the model

`...` Additional arguments passed to the test

*Returns:* Result of the test

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
TestFactory$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## test_factory *Shared instance of the test factory*

### Description

Users can register new tests via this object. Built-in tests are pre-registered at package load time.

### Details

Deprecated since 0.14.0, with TestFactory: it leaves the exports in 1.0.0, and as an object it cannot warn when used. Use `registerDiagnostic()` and `runHeteroTests()` instead.

Its `"szroeter"` entry is `performSzroeterTest()`, which needs the column that orders the observations, given as `order_by`: `test_factory$run_test("szroeter", model, data, order_by = "x")`. Without `order_by` it refuses the call and asks for it, since 0.16.0; before, it stopped with "argument "order_by" is missing, with no default".

## test_results *What every test returns*

### Description

Every exported `perform*Test()` function returns a base-R `htest` object, so a result prints, subsets and composes like the result of `stats::t.test()` or `lmtest::bptest()`. This page lists every field such a result can hold: the five every test returns, the two optional `htest` fields some return, and the fields a few tests add. '`tests/testthat/test-return-contract.R`' checks every exported test against it.

### Fields of every test

`statistic` the test statistic: a numeric value of length one, named after the statistic (`"X-squared"`, `"F"`, `"t"`, `"GQ"`, `"LM"`, `"z"`, ...).

`parameter` the parameters of the null distribution: a numeric vector whose elements are named, usually degrees of freedom (`"df"`, or `"df1"` and `"df2"`). The resampling tests store the number of replications here (`"B"`, `"permutations"`), and the tests on ordered observations the sizes they use (`"n"`, `"m"`). It is `NULL` for `performPesaranTest()`, whose statistic is referred to the standard normal distribution.

`p.value` the p-value: a single number between 0 and 1.

`method` a character string naming the test. When an argument switches between two statistics, such as `studentize` in `performHarveyTest()`, the string says which one was computed.

`data.name` a character string describing what was tested: the formula of the model, followed for some tests by the variance regressors. For `performBoxMTest()`, which tests data rather than a model, it names the data and the grouping.

### Optional htest fields

`alternative` a character string describing the alternative hypothesis, returned by the tests that have a direction or a stated alternative.

`estimate` estimates that accompany the statistic: a named numeric vector (`performSzroeterTest()`, `performSpearmanTest()`, `performVarianceFormTest()`), or, in `performQuantileRegressionTest()`, a matrix of slope estimates with one column per quantile and named rows and columns.

### Fields some tests add

`p.value_bootstrap` `performWhiteTestRobust()` and `performBPTestRobust()` with `bootstrap = TRUE`: the bootstrap p-value.

`robust_details` `performWhiteTestRobust()` and `performBPTestRobust()`: a list with the confidence interval, effect size, bootstrap results, power and sample size, and `method` or `studentized`, the setting that chose the statistic.

`boot_statistics` `performWhiteTestBootstrap()`: the bootstrap replicates of the statistic.

`bootstrap` `performWildBootstrapTest()`: a list with the replicates, the number of usable ones, the multiplier distribution and the observed statistic.

`permutation` `performRankPermutationTest()`: a list with the replicates, the number of usable ones and the ordering variable.

`projection` `performHighDimensionalTest()`: a list with the number of principal components kept, the share of the variance they explain and the threshold that chose them.

`variance_function` `performVarianceFormTest()`: a list with the form of the variance function, its regressors and the added terms that were tested.

`quantiles`, `se_type`, `iid` `performQuantileRegressionTest()`: the quantiles compared and the settings of the standard errors.

`chunks_processed`, `chunk_size`, `r_squared` `performWhiteTestStreaming()`, `performBPTestStreaming()` and `performKoenkerTestStreaming()`: how the data were read and the $R^2$ of the auxiliary regression.

`res` `performSpatialHeteroTest()`: the Monte Carlo replicates of Moran's I from `spdep::moran.mc()`, whose class `"mc.sim"` the result keeps.

### Attributes

`performWhiteTest()` and `performWhiteTestRobust()` attach the attributes `auxiliary_regressors`, `original_regressors`, `cross_products_included` and `r_squared_auxiliary`, which describe the auxiliary regression. `runHeteroTests()` adds attributes of its own to each result it returns.

### Inputs every test refuses

Since 0.15.0 every test on a model refuses four inputs, each with one message: a `glm` that is not a least-squares fit, that is, of any family but `gaussian` or with any link but the identity ("needs a model fitted by stats::lm()"); a model with a coefficient that is not finite, infinite, `NaN`, or `NA` without the fit being rank-deficient ("Model coefficients must be finite"); `data` that lack rows the model was fitted to; and `data` that lack a variable of the model ("The following variables are missing from"). Until 0.16.0 the `NA` of an aliased term was refused too; since then a rank-deficient fit is tested on the

estimable part of its design (see *Supported models*). The model frame is accepted as `data` as it is. A `glm` with the Gaussian family and the identity link is accepted by every `perform*Test()` function and gives the result of the `lm` up to rounding error, when the two decide the rank of the design alike (see *Supported models*); `performWhiteTestStreaming()`, `performHarrisonMcCabeTest()` and `performVarianceFormTest()` refused every `glm` until 0.16.0, and `bootstrapTestStatistic()` until 0.17.0, which bootstraps it as the `lm` it is. With positive prior weights it gives the result of `lm(..., weights = w)`: `performGQTest()`, `performRESETTest()`, `performWhiteTestBootstrap()`, `performWildBootstrapTest()` and `performHarrisonMcCabeTest()`, which read the weights of an `lm`, read the prior weights of the `glm`, and `performQuantileRegressionTest()`, which refuses a weighted `lm`, and `performVarianceFormTest()`, which refuses weights that did not come from `fitWLS()`, refuse it with the same message. With a zero prior weight it is refused as the `lm` with a zero weight is ("The model was fitted with zero or non-finite weights"), since 0.16.0. A `survey::svyglm()` fit is refused by every test ("The model is a `survey::svyglm()` fit"). `tests/testthat/test-input-val` gives every test these inputs. Since 0.17.0 a `MASS::rlm()` fit is refused by every test too, with class `heteroTests_model_context_error` (see *Supported models*), which 'tests/testthat/test-model-context.R' checks.

The tests that read `data` also refuse `data` whose rows cannot be confirmed to hold the observations of the fit ("cannot match the rows of `data` to the observations of the model"). They match the rows of `data` to the residuals by row name, and a tibble, a data.table or a data frame whose row names were reset, as by `dplyr::arrange()`, has row names that only number its rows. Before a test reads `data`, the rows the row names pick are compared with the model frame of the fit, row by row: every column `data` holds under its own name, value by value, and the others (`log(x)`, `poly(x, 2)`, `offset(o)`, the weights and an offset given as arguments) computed from `data` as the fit computed them. A term computed from the rows it is given, such as `I(x - mean(x))`, and a column that names a variable `data` does not hold, are not compared; for a term of variables of `data`, the compared columns must then tell the rows of the fit apart. Data that cannot be confirmed so are refused, even the data of the fit: when no column can be compared, as with `lm(d$y ~ d$x)`, or when the compared columns tie on rows that differ in such a term. Pass the full data the model was fitted on, or the variables as columns of `data`. Data without row names of their own in another order than the fit, or with rows in front of those of the fit, are refused, as are repeated row names, and a model frame rebuilt for a fit with `model = FALSE` from data that changed after the fit. Data with row names of their own are matched by name, in any order; until 0.16.0 `performPesaranTest()`, `performBPRandomEffectsTest()` and `performScatterDiagnostic()` read `data` by position, and refused such data in another order than the fit. `tests/testthat/test-data-alignment-guard.R` checks these refusals.

Since 0.16.0 `performWhiteTest()`, `performBPTest()`, `performKoenkerTest()`, their streaming forms, `performWhiteTestStreaming()`, `performBPTestStreaming()` and `performKoenkerTestStreaming()`, which read the design of the fit chunk by chunk where each chunk built its own from the rows of `data`, `performRankPermutationTest()`, `performHighDimensionalTest()`, the studentized Breusch-Pagan statistic (`performStudentizedBPTest()`, `performBPTestRobust()` with `studentize = TRUE`, the registered `"student_bp"`), `performRESETTest()`, `performGQTest()`, `performQuantileRegressionTest()`, `performGlejserTest()`, `performParkTest()`, `performSzroeterTest()` with the registered `"szroeter"`, `performHarrisonMcCabeTest()`, `performHartleyFmaxTest()`, `performSpatialHeteroTest()`, the five group tests, `performLeveneTest()`, `performBrownForsytheTest()`, `performFlignerKilleenTest()`, `performBartlettTest()` and `performOBrienTest()`, the two panel tests, `performPesaranTest()` and `performBPRandomEffectsTest()`, and so `runPanelTests()`, and `performScatterDiagnostic()`, and the tests computed from the model alone, `performSpearmanTest()`, `performHarveyTest()`, `performArchLMTest()`, `performMcLeodLiTest()`, `performCookWeisbergTest()` and `performNCVTest()`,

performSpreadLevelTest(), performDavidianCarrollTest() and performVarianceFormTest(), with fitWLS(), read the response, the design, the residuals, the weights and the offset from the fit, through the model context of the package, and read from data only the rows the comparison above confirmed; the other tests follow in later steps (the design page of the site, https://diogoribeiro7.github.io/heteroTests/design/model-context/, sets out the plan). RESET, the tests computed from the model alone and fitWLS() without var_formula read nothing from data, and match it as the other tests do, so that every test takes and refuses the same data; until 0.16.0 they gave the value of the fit on numbered data in another order, rows in front of those of the fit and repeated row names, and refused numbered data in the order of the fit whose row numbers are not the fit's (fitWLS() ignored data). A second formula, the var_formula of the Cook-Weisberg test, of the variance form test and of fitWLS(), and the against of the variance form test, is evaluated on the confirmed rows of data, or, without data, on the model frame; a variable that neither holds is refused, where it was found in the workspace. The Goldfeld-Quandt, Szroeter and Harrison-McCabe tests read their order_by, Glejser's and Park's their variable and Hartley's and the group tests their group from the confirmed rows of data, or, without data, from the model frame when it holds the variable under its own name ("The following variables are missing from the model frame of model" otherwise). Hartley's test refuses a group that is missing at an observation of the fit; the group tests leave such observations out, with the warning "Removed N observations due to missing values in ...", as before 0.16.0. The panel tests read their id and time, and the scatter diagnostic its vars, from the confirmed rows of data; without data the panel tests read them from the model frame, when it holds them under their own names, and the scatter diagnostic needs data. Pesaran's test refuses a pair of id and time that holds two observations of the fit and a unit or a period missing at an observation of the fit, and computes the statistic of an unbalanced panel, whose units are not all observed at the same periods, from the periods each pair of units shares, as plm::pcdtest() does. The registered "szroeter" orders by the fitted values, which it computes from the fit, and the spatial test reads no column of data: its listw lists the observations of the fit in its order. Harrison-McCabe takes the rank of the fit, decided at the fit's tolerance (1e-7 for an lm, 1e-11 for a glm). These tests match numbered rows by the row numbers of the fit, or, when data holds one row per observation and its numbers are not those of the fit, in the order of the fit, once those rows reproduce the model frame; they drop nothing before the match, so data that hold the rows a fit with na.omit dropped, or only the rows it kept, numbered again, are accepted. Since 0.17.0 a fit with na.action = na.exclude is tested as the same fit with na.omit, which drops the same rows, with the same values, refusals and warnings: before, the tests refused it ("Model residuals must be finite"), RESET stopped on it ("NA/NaN/Inf in 'x'"), and the panel tests and the scatter diagnostic read its residuals, padded with NA at the rows it dropped, against the rows of data by position, which gave a statistic that was NA, or computed with the NA. An output with one value per row, as the estimated variances fitWLS() attaches to its fit, is padded with NA at the rows the fit dropped, as residuals() and weights() of such a fit are. Named data in any order, numbered data in the order of the fit: numbered data in another order are refused, as by the other tests. They need no data for a model that transforms its variables, has an offset() term, or takes its weights or offset from a column of the data, and accept its model frame as data. They also refuse a multivariate lm ("needs a model with one response"), a fit whose coefficients, residuals or weights were changed after it was fitted ("does not reproduce its residuals from its design and coefficients", "The weights of the model are not those its coefficients were fitted with"), a fit whose terms do not rebuild the design of its coefficients, as a fitWLS() fit of an mgcv::gam() fit with smooth terms (the first of these messages, since 0.17.0), and a fit with model = FALSE whose model frame can no longer be rebuilt, unless data gives its response and fitted values, on which it is then tested. The tests that fit the model again by least squares, or whose null distribution is derived for

the least-squares residuals of its design, refuse an `mgcv::gam()` fit with penalised smooth terms, which is not the least-squares fit of its design, and the others test it on that design, the basis of its smooths, since 0.17.0 (see *Supported models* below). The residuals of a `gam` carry no names, and these tests match `data` to its observations by the row names of its model frame; the tests that are not built on the context yet and read a column of `data` match them by position. The check of an edited fit is a guard against accidental edits, not a proof that a fit is unedited: an edited coefficient leaves the residuals unchanged, so it is seen only at `1e-7` of the largest term of the response, the design times the coefficients and the residuals, and an edit of 0.5 goes undetected once the response is above about `1e7`. These refusals have class `heteroTests_model_context_error`, which `runHeteroTests()` and `runDiagnostics()` keep, as they keep that of a refusal of `data`, without running another test under the name of the one refused; since 0.17.0 a grouped run stops with such a refusal in a group too, where the group recorded it as a failed test of `NA`. A test that reads a column of `data` the model frame does not hold, as the rank permutation, Goldfeld-Quandt, Szroeter and Harrison-McCabe tests do with their `order_by`, Glejser's and Park's with their `variable`, Hartley's and the group tests with their `group`, the panel tests with their `id` and `time` and the scatter diagnostic with its `vars`, refuses numbered rows taken in the order of the fit if two rows of the fit tie on every compared column. On a call with more than one defect, the first one reported may differ from before 0.16.0, since the rows of `data` are now matched before the checks of the model. `tests/testthat/test-model-context.R` checks them.

### Supported models

The tests examine the residuals of a least-squares fit for a variance that is not constant, so they accept the models whose residuals are those of a least-squares fit, and refuse the others early, with the messages above. Which test accepts which class, measured by running every test on each, is the table of the *Supported models* page of the documentation site; `tests/testthat/test-model-eligibility.R` and `test-input-validation.R` check it.

- An `lm` is accepted by every test.
- An `lm(..., weights = w)` is accepted: `lm()` states that the error variance is $\sigma^2/w_i$, and the tests read the Pearson residuals $\sqrt{w_i}\,e_i$, which have constant variance when the weights are right. `performQuantileRegressionTest()`, which refits without them, refuses it, and `performVarianceFormTest()` refuses weights that did not come from `fitWLS()`.
- A zero weight: the observation is not part of the fit, and its residual is an artificial zero that a test would count. Refused by every test but `performRESETTest()`, an F test of two nested weighted least-squares fits, in which a zero weight drops the observation from both; it gives the value of the fit to the observations with positive weight. Keeping that exception was decided on 2026-10-17.
- An offset, given as `offset = o` or as an `offset()` term: the residuals of the fit include it, and the tests that read them account for it. Four left it out of a refit and returned a value that was not that of the model; they refused it in the development of 0.16.0 (decided 2026-10-17), and since the offset fix of 0.16.0 they read the offset from the fit and carry it in every refit: `performGQTest()`, `performQuantileRegressionTest()` and the studentized Breusch-Pagan statistic (`performStudentizedBPTest()`, `performBPTestRobust()` with `studentize = TRUE`, the registered `"student_bp"`) give the value of the model fitted to the response minus the offset, and `performRESETTest()` the nested F test of the model with its offset against the model augmented with the powers of its fitted values, $X\hat{\beta} + o$ (decided 2026-10-18).

- A `glm` with the Gaussian family and the identity link is a least-squares fit. With prior weights $w_i$ it is fitted by maximum likelihood, which for normal errors of variance $\phi/w_i$ maximises $-\sum_i w_i(y_i - x_i^\top\beta)^2/(2\phi)$: weighted least squares. Iteratively reweighted least squares, the algorithm of `glm()`, then has the working response $y_i$ and the working weights $w_i$ whatever the current estimate, so its first step reaches the solution of `lm(..., weights = w)` (Nelder and Wedderburn, 1972; McCullagh and Nelder, 1989, sections 2.5 and 3.6). It is Fisher scoring, and for the identity link, the canonical link of the Gaussian family, the information does not depend on $y$, so scoring is Newton-Raphson, which converges in one step when the log-likelihood is quadratic in $\beta$ (Green, 1984, section 1.2). Its design, coefficients, fitted values, residuals and weights are those of the `lm`, up to rounding, so a test whose statistic and null distribution are functions of those, under the assumptions of the linear model, which the `glm` states, gives the result of the `lm`: every `perform*Test()` function does, weighted or not (‘`tests/testthat/test-input-validation.R`’ and ‘`test-model-eligibility.R`’ compare them). That holds when the two decide the rank of the design alike: `lm()` aliases a column with the tolerance `1e-7` of its QR decomposition, `glm.fit()` with `min(1e-7, epsilon / 1000)`, `1e-11` by default, so on a column within that margin of a combination of the others, such as `x3 = x1 + 1e-9 * rnorm(n)`, the `glm` keeps it and gets the values of `lm(..., tol = 1e-11)`, not those of the default `lm`. `performWhiteTestStreaming()`, `performHarrisonMcCabeTest()` and `performVarianceFormTest()` refused every `glm` until 0.16.0, and `fitWLS()` refits it as that `lm` since then. `bootstrapTestStatistic()` refused every `glm` until 0.17.0, as it refits each replicate by least squares, which would drop a family and a link; it bootstraps a Gaussian `glm` with the identity link as that `lm`, with its prior weights, and refuses any other `glm`, as before. It refused every `mgcv::gam()` fit until 0.17.0, and refuses one with penalised smooth terms since then (see the item on the `gam` below). The checks of the model read its R-squared, which `summary()` of a `glm` does not report, as that of the `lm`: `summary.lm()` applied to the `glm`, whose fitted values, residuals, weights and offset are those of the `lm`, so that a near-perfect fit is refused as the `lm` is, since 0.16.0. The *Supported models* page of the documentation site sets out the argument; the references are below, and the section numbers of McCullagh and Nelder (1989) are those of its table of contents. The tests read its prior weights, `model$prior.weights`, and never `model$weights`, which holds the working weights of the last iteration of iteratively reweighted least squares, $w_i(d\mu_i/d\eta_i)^2/V(\mu_i)$. For this family and link the two are equal up to rounding; for any other they depend on the fitted means. A `glm` fitted with weights that are all 1 has the `prior.weights` of one fitted without weights, so the tests take it for an unweighted fit, where `lm(..., weights = rep(1, n))` is a weighted one: `performVarianceFormTest()` and `performQuantileRegressionTest()`, which refuse a weighted fit, accept the `glm` and refuse the `lm`.
- Any other `glm`, such as a binomial or a Poisson one, is refused: its residuals have a variance that depends on the mean by construction, so a test of constant variance tests nothing the help pages describe.
- A `survey::svyglm()` fit is refused: its residuals are scaled by the sampling weights, which are not precision weights. See `runSurveyHeteroTests()`.
- Panel models: `performPesaranTest()` and `performBPRandomEffectsTest()`, and so `runPanelTests()`, take a pooled `lm` and the columns of `data` that name the units and the periods. A `plm::plm()` fit is not an `lm`: every test refuses it, as it refuses anything but an `lm` or a `glm`, with one message, which names the class received and points to the pooled `lm()`.
- A rank-deficient fit: a term that is a linear combination of others, or the dummy of a cell of an interaction that no observation falls in, leaves an `NA` coefficient. Its residuals and fitted values are the projection on the column space of the design, which the aliased columns do not

change. Since 0.16.0 every test that accepts the model without the aliased columns accepts it, computed on the estimable part of the design: the columns that were not aliased, with degrees of freedom from the rank, so that it gives the statistic and p-value of the model fitted without them. A test that refuses that model refuses the rank-deficient one with the same message. The model without them is the one `lm()` keeps: its decomposition keeps a column unless it is a combination of the columns before it, so of an aliased set the column that comes last, in the order of the terms, is left out. Most statistics depend only on the space the columns span, and are the same whichever column is left out. Two depend on which: `performWhiteTest()`, and its robust, streaming and bootstrap forms, when the cross products are left out (`cross_products = FALSE`, `method = "reduced"`, or more regressors than `max_interactions`), as the squares are those of the columns kept; and `performHighDimensionalTest()` when it keeps fewer principal components than columns. With `form = "power"`, `performVarianceFormTest()` takes the logarithms of every column of the design, the aliased ones included, so its value is not that of the model without them (and does not depend on the order of the terms); with the exponential form, the default, it is.

- A `fitWLS()` fit is a weighted `lm`, with estimated weights: accepted, but the reference distributions take the weights as known (`performVarianceFormTest()` accounts for the estimation).
- A `MASS::rlm()` fit, such as `fitRobust()` returns, is an `lm` by class and an M-estimate, whose residuals are not least-squares residuals. Since 0.17.0 every test refuses it, with an error of class `heteroTests_model_context_error` that names the `lm()` fit of the same formula to test instead, and so do `fitWLS()`, the registered names and the runners, which keep the refusal final. Until then 37 tests took it for an `lm` and tested its M-estimation residuals as if they were least-squares ones, the studentized Breusch-Pagan statistic tested the least-squares refit of its design, and `performQuantileRegressionTest()` and `performVarianceFormTest()` refused it as a weighted fit, its case weights, all 1 unless given, being read as precision weights.
- An `mgcv::gam()` fit with the Gaussian family and the identity link, a `glm` by class, is tested on its stored design, `model.matrix()`: its parametric columns and the basis of its smooths. A smooth term is penalised unless it is fixed, `s(x, fx = TRUE)`, and the residuals of a penalised fit are not those of the least-squares fit of that design. Since 0.17.0 the tests that read only its residuals, its fitted values, columns of `data` and the columns of the design as auxiliary regressors test a penalised fit, the streaming tests on the basis too, as the exact tests do, where they read the variables of the formula. The tests that fit it again by least squares, or whose null distribution is derived for the least-squares residuals of its design, refuse it with class `heteroTests_model_context_error`, and the runners keep the refusal final: `performGQTest()`, `performRESETTest()`, `performQuantileRegressionTest()`, the studentized Breusch-Pagan statistic, `performWhiteTestBootstrap()`, `performWildBootstrapTest()`, `bootstrapTestStatistic()` and the robust forms with `bootstrap = TRUE`, `performVarianceFormTest()` and `fitWLS()`, which refitted it without its penalty, and `performHarrisonMcCabeTest()` and `performSzroeterTest()`, which compared its residuals with the null distribution of least-squares ones. A `gam` that is the least-squares fit of its design, without smooth terms or with fixed ones, is tested by every test as the `lm()` of that design, the refitting and resampling tests included, where the fixed smooths were refused as penalised ones. Whether a `gam` is that fit is decided from the fit: its design refitted by least squares, with its weights and offset, gives its residuals to `1e-8` of the largest residual of that refit, plus `1e3` times the machine epsilon of the largest value of the response and of the offset, the rounding of a fit on that scale, so that an offset or a trend does not change the verdict, up to the scale at which that rounding hides a penalised fit (the largest value of the response or of the offset about `1e12` times its residuals). A `gam` without a penalised term that fails the check is refused as one whose residuals are not

those of that fit, not as a penalised one: mgcv's own fit is not the least-squares fit of an ill-conditioned design, as one with a regressor far from zero (centre the regressor, or fit the model with `lm()`), or the fit was changed after it was fitted. Decisions 37 and 38 of the design of the model context (decided on 2026-10-29) set this out.

`checkModel()` checks the class of a model only, `lm` or `glm`; it does not check whether a test accepts it.

## References

Green, P. J. (1984). Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *Journal of the Royal Statistical Society, Series B, 46*(2), 149–192. doi:10.1111/j.25176161.1984.tb01288.x

McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models* (2nd ed.). Chapman & Hall. The section numbers cited are those of its table of contents.

Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society, Series A, 135*(3), 370–384. doi:10.2307/2344614

## See Also

`runHeteroTests`, `t.test` for the `htest` class

---

# `theme_hetero` *Heteroscedasticity diagnostic theme*

---

## Description

Applies a light-minimal theme with subtle gridlines and bold titles to maintain visual consistency across diagnostic plots.

## Usage

```
theme_hetero(base_size = 12, base_family = "")
```

## Arguments

| | |
|---|---|
| `base_size` | Base font size. |
| `base_family` | Base font family. |

## Value

A ggplot2::theme object.

## Examples

```
theme_hetero()
```

`tidy.hetero_test` *broom methods for heteroscedasticity diagnostics*

## Description

Tidy, glance and augment methods integrating heteroscedasticity diagnostics with the broom generics from the **generics** package.

## Usage

```
## S3 method for class 'hetero_test'
tidy(x, ...)
## S3 method for class 'hetero_test'
glance(x, ...)
## S3 method for class 'hetero_test'
augment(x, data = NULL, ...)
## S3 method for class 'hetero_test_suite'
tidy(x, ...)
## S3 method for class 'hetero_test_suite'
glance(x, ...)
## S3 method for class 'hetero_test_suite'
augment(x, data = NULL, ...)
## S3 method for class 'hetero_grouped_suite'
tidy(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Heteroscedasticity diagnostic result(s) returned by `runHeteroTests`. |
| `data` | Optional data frame used to augment diagnostics; defaults to the data embedded within the fitted model when available. The rows of `data` are matched to the observations of the fit as the tests match them, by row name in any order, or for data that only number their rows by the row numbers of the fit or in its order, and bound as broom binds them, in the order of `data`, each row with its own fitted value and residual: on a fit without `na.action = na.exclude` the rows the fit used, so the full data of a fit that dropped rows give those rows; on a fit with `na.exclude` those rows and the rows it dropped, with `NA` there, every row of its full data. Given the rows of its observations only, as its model frame without `data`, a fit with `na.exclude` binds them in the order of the fit with a row of `NA` at each row it dropped. Data whose rows cannot be shown to hold the observations are refused (see `test_results`). The data of a tidymodels workflow, or of a parsnip fit of `fit_xy()`, are mapped to the variables of its engine as `runHeteroTests()` maps them, and each mapped row is traced to the row of `data` it was made from: the data of a preprocessing that drops or adds rows of new data, such as `step_filter()` with `skip = FALSE`, or reorders them, such as `step_arrange()`, are refused, since it does not say which row of `data` each observation comes from, and so are those of a step that makes a row from other rows, such as `step_mutate(cs = cumsum(x))`, whose values 0.16.0 |

bound right in the order of the fit. Before 0.17.0 the rows were bound row by row: named data in another order than the fit had the values of other observations bound to them, and the full data of a fit with `na.omit` could not be bound.

`...` Passed through for compatibility.

## Value

A data frame summarising the diagnostic(s), or an augmented data set containing fitted values and residuals alongside the original predictors.

`tidy` returns one row per test. `parameter` is the first parameter of the test and `parameter2` the second, for a test that has two (the denominator degrees of freedom of an F statistic, or the number of bootstrap replications); it is `NA` otherwise. `estimate` is given when the test reports exactly one estimate. Up to 0.12.0 a test with two parameters was returned as two rows.

## See Also

`generics::tidy`, `generics::glance`, `generics::augment`

## Examples

```
if (requireNamespace("generics", quietly = TRUE)) {
  data(mtcars)
  fit <- lm(mpg ~ wt + qsec, data = mtcars)
  suite <- runHeteroTests(fit, mtcars, tests = c("white", "breusch_pagan"))
  generics::tidy(suite)
}
```

---

# validateTestInputs *Validate model and data inputs before running diagnostics (deprecated)*

---

## Description

Deprecated since 0.14.0: an internal helper of the package, which leaves the exports in 1.0.0. It still works and warns.

## Usage

```
validateTestInputs(model, data, test_name, min_obs = 10)
```

## Arguments

| | |
|---|---|
| `model` | A fitted model created by `stats::lm()` or `stats::glm()`. |
| `data` | A `data.frame` containing the variables used to fit `model`. |
| `test_name` | Character scalar naming the diagnostic that is about to run; included in error messages for clarity. |
| `min_obs` | Non-negative integer giving the minimum sample size accepted by the diagnostic. Defaults to `10`. |

## Details

Provides the compatibility wrapper used by the public testing interface to ensure that fitted-model objects and their associated data satisfy minimal quality requirements. The helper guards against the most common issues that invalidate heteroscedasticity tests and produces actionable error messages that reference the calling diagnostic.

The routine performs four layers of validation:

1. confirm that `model` inherits from `lm` or `glm` and that its coefficients and residuals are finite;
2. verify that `data` is a `data.frame` with at least `min_obs` rows;
3. ensure the residual vector is available, finite, and aligned with the supplied data;
4. emit warnings when studentised residuals exceed five standard deviations in absolute value or when the dataset is very large (more than 10,000 observations).

Results are cached (when the **digest** package is installed) so repeated calls with unchanged inputs return immediately.

## Value

Invisibly returns `TRUE` when validation succeeds. Execution stops with an informative error when any check fails.

## References

Fox, J. (2015). *Applied Regression Analysis and Generalized Linear Models* (3rd ed.). SAGE.

Belsley, D. A., Kuh, E., & Welsch, R. E. (1980). *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Wiley.

## See Also

`checkModel()`, `rvalidateModelInputs()`, `rvalidateTestRequirements()`

## Examples

```
data(mtcars)
lm_fit <- stats::lm(mpg ~ wt + hp, data = mtcars)
validateTestInputs(lm_fit, mtcars, "white")


noisy <- mtcars
noisy$mpg[1] <- 100
outlier_fit <- stats::lm(mpg ~ wt + hp, data = noisy)
validateTestInputs(outlier_fit, noisy, "white")
```

# Index


∗ **datasets**
.diagnostic_registry, 4
modern_diagnostics, 33
.diagnostic_registry, 4

additional_tests, 5
analyseDatasetCharacteristics *(*generateHeteroRecommendations*)*, 27
analyzeMLResiduals, 5, *19*
anova(), *151*
augment.hetero_test *(*tidy.hetero_test*)*, 164
augment.hetero_test_suite *(*tidy.hetero_test*)*, 164
autoCompareRemediations, 6
autoCompareRemediations(), *22*
autoplot.hetero_grouped_suite *(*autoplot.hetero_test_suite*)*, 6
autoplot.hetero_test_suite, 6
autoTransform, 7

base::data.frame, *23*, *98*, *130*, *142*
bootstrap_methods, 13
bootstrapTestStatistic, 8, *158*, *161, 162*
bootstrapTestStatistic(), *25*, *43*, *111*, *130*
buildDiagnosticDecisionTree *(*generateHeteroRecommendations*)*, 27

cachedTest, 13, *17*, *41*, *69*, *105*, *127*, *134*
cachedTest(), *18*, *136*, *152*
checkData, 15
checkModel, 15, *163*
checkModel(), *17*, *166*
checkModelEnhanced, 16
clearAnalysisCache, 17, *134*
clearAnalysisCache(), *14*, *18*
clearTestCache, *17, 18*, 18
clearTestCache(), *14, 15*
coef.summary.hetero_grouped_suite *(*summary.hetero_test_suite*)*, 150
coef.summary.hetero_test_suite *(*summary.hetero_test_suite*)*, 150
compareModelDiagnostics, *5*, 19
compareTestResults, 20
composeDiagnosticNarrative *(*generateHeteroRecommendations*)*, 27

diagnostic_data, 20
downloadTeachingData, 21

fitRobust, 22, *162*
fitWLS, 23, *42*, *70*, *75*, *106*, *133*, *158–162*
fitWLS(), *9, 10*, *14*, *22*, *43*, *92*, *98*, *100*, *107, 108*, *111*, *116, 117*, *130*, *152*
fligner.test, *50–52*
format.pval(), *151*

generate_benchmark_report *(*run_benchmark_suite*)*, 140
generateDiagnosticReport, 26
generateHeteroRecommendations, 27
generics::augment(), *152*
generics::glance(), *152, 153*
generics::tidy(), *152, 153*
ggplot2::scale_colour_manual(), *143*
ggplot2::theme, *163*
glance.hetero_test *(*tidy.hetero_test*)*, 164
glance.hetero_test_suite *(*tidy.hetero_test*)*, 164

hetero_data, 29
hetero_grouped_suite, *7*, *135*, *150*
hetero_grouped_suite *(*hetero_test_suite*)*, 30

hetero_test_suite, *7*, 30, *135*, *139*, *150*, *152*
HeteroDiagnostic, 29
ht_clear_log_history *(*ht_set_log_level*)*, 30
ht_enable_log_capture *(*ht_set_log_level*)*, 30
ht_log_history *(*ht_set_log_level*)*, 30
ht_set_log_level, 30

interpretHeteroTestResults *(*generateHeteroRecommendations*)*, *27*

launchDiagnosticDashboard, 31
listDiagnostics, 32, *127*, *133*
listDiagnostics(), *13*
lmtest::bptest(), *93*
lmtest::hmctest(), *59, 60*

modern_diagnostics, 33

parallel::makeCluster(), *12*, *107*, *136*
parallel::mclapply(), *12*, *107*
parallel::nextRNGStream(), *12*, *108*
performArchLMTest, 33, *158*
performBartlettTest, 34, *158*
performBoxMTest, 36, *156*
performBPRandomEffectsTest, 38, *158*, *161*
performBPTest, 39, *44–46*, *49*, *65*, *76*, *134*, *135*, *158*
performBPTest(), *42*, *44*, *94*
performBPTestRobust, *41*, 42, *69*, *157, 158*, *160*
performBPTestRobust(), *11*, *94*, *130*
performBPTestStreaming, *40, 41*, 44, *134*, *157, 158*
performBreuschPaganTest *(*performBPTest*)*, 39
performBrownForsytheTest, 46, *158*
performCookWeisbergTest, 47, *75, 76*, *158*
performDavidianCarrollTest, 49, *159*
performFlignerKilleenTest, 50, *158*
performGlejserTest, 53, *65*, *158*
performGlejserTest(), *60, 61*, *96*
performGQTest, 56, *158*, *160*, *162*
performGQTest(), *59–61*, *97*
performHarrisonMcCabeTest, 58, *158*, *161, 162*
performHarrisonMcCabeTest(), *97*
performHartleyFmaxTest, 62, *158*
performHarveyTest, *55*, 63, *156*, *158*
performHarveyTest(), *43*, *100*
performHighDimensionalTest, 65, *157, 158*, *162*
performInfluenceDiagnostics, 67
performKoenkerTest, *40*, *42*, *48, 49*, *55*, 67, *71*, *75, 76*, *93*, *106*, *134, 135*, *158*
performKoenkerTest(), *10, 11*, *14*, *44*, *92–94*, *97*, *99*
performKoenkerTestStreaming, *68, 69*, 70, *134*, *157, 158*
performLeveneTest, 72, *158*
performMcLeodLiTest, 73, *158*
performNCVTest, *47–49*, 74, *158*
performOBrienTest, 76, *158*
performParkTest, *55*, *65*, 78, *158*
performParkTest(), *100*
performPesaranTest, 79, *138*, *156*, *158*, *161*
performQuantileRegressionTest, 81, *157, 158*, *160–162*
performRankPermutationTest, 83, *157, 158*
performRESETTest, 85, *158*, *160*, *162*
performScatterDiagnostic, 86, *158*
performSpatialHeteroTest, 88, *157, 158*
performSpearmanTest, 89, *157, 158*
performSpreadLevelTest, 91, *159*
performStudentizedBPTest, *69*, 92, *158*, *160*
performSzroeterTest, 94, *157, 158*, *162*
performSzroeterTest(), *60, 61*, *156*
performVarianceFormTest, *70*, 97, *157–162*
performVarianceFormTest(), *24, 25*, *118*
performVIFDiagnostic, 101
performVIFDiagnostic(), *16, 17*
performWhiteTest, 102, *114*, *134, 135*, *157, 158*, *162*
performWhiteTest(), *108–110*, *112–115*
performWhiteTestBootstrap, *103–105*, 106, *133*, *157, 158*, *162*
performWhiteTestBootstrap(), *25*, *112*
performWhiteTestRobust, *103*, *105*, 110, *157*
performWhiteTestRobust(), *11*, *110*, *115*, *130*
performWhiteTestStreaming, *45*, *71*, *102–105*, 112, *134*, *157, 158*, *161*
performWildBootstrapTest, *69*, 115, *133*,

*157*, *158*, *162*
performWildBootstrapTest(), *10*, *25*, *109*
plot.HeteroDiagnostic, 119
plot.power_analysis, 119
plotBeforeAfter, 120
plotBubbleVariance, 121
plotDiagnosticSuite, 121
plotDiagnosticSuiteEnhanced, 122
plotResidualDensity, 122
plotResidualQQ, 123
plotResidualsFitted, 123
plotResidualsFitted(), *121*
plotResidualsFittedEnhanced, 124
plotSpreadLevel, 125
print.hetero_grouped_suite *(*summary.hetero_test_suite*)*, 150
print.hetero_recommendation_report *(*generateHeteroRecommendations*)*, 27
print.hetero_test_suite *(*summary.hetero_test_suite*)*, 150
print.htest_enhanced, 125
print.remediation_suggestions *(*suggestRemediation*)*, 148
print.summary.hetero_grouped_suite *(*summary.hetero_test_suite*)*, 150
print.summary.hetero_test_suite *(*summary.hetero_test_suite*)*, 150
print.summary.lm(), *151*
printCoefmat(), *151*, *153*

qr, *103*
qr(), *113*, *114*

rbootstrap_test_statistic *(*bootstrapTestStatistic*)*, 8
rcalculateEffectSize, 126
recommendRemediationStrategies *(*generateHeteroRecommendations*)*, 27
registerDiagnostic, 127
registerDiagnostic(), *13*, *32*, *151*, *154*, *156*
registerPlot, 127
restimate_test_power, 128
robust_implementations, 129
rrunAdvancedDiagnostics *(*runAdvancedDiagnostics*)*, 129
run_benchmark_suite, 140
runAdvancedDiagnostics, 129
runDiagnosticPlots, 130
runDiagnostics, *41*, *69*, *105*, 131, *160*
runDiagnostics(), *22*
runHeteroTests, *17–19*, *41*, *69*, *103–105*, *127*, *132*, 132, *138*, *139*, *149*, *157*, *160*, *163*, *164*
runHeteroTests(), *14*, *15*, *18*, *20*, *22*, *32*, *149*, *151–154*, *156*
runHeteroTestsParallel, 136
runMultivariateTests, 137
runMultivariateTests(), *32*
runPanelTests, 137, *140*, *158*, *161*
runSurveyHeteroTests, *30*, 138, *161*
runSurveyHeteroTests(), *152*
runTimeSeriesTests, *138*, 139
rvalidate_against_reference, 142
rvalidateModelInputs(), *93*, *166*
rvalidateTestRequirements(), *93*, *166*

scale_colour_hetero_diagnostic, 143
scale_fill_hetero_diagnostic *(*scale_colour_hetero_diagnostic*)*, 143
sigma_exp_decay *(*simulate_hetero*)*, 144
sigma_exponential *(*simulate_hetero*)*, 144
sigma_gaussian_peak *(*simulate_hetero*)*, 144
sigma_group *(*simulate_hetero*)*, 144
sigma_inverse *(*simulate_hetero*)*, 144
sigma_linear *(*simulate_hetero*)*, 144
sigma_logistic *(*simulate_hetero*)*, 144
sigma_multiplicative *(*simulate_hetero*)*, 144
sigma_piecewise *(*simulate_hetero*)*, 144
sigma_piecewise_linear *(*simulate_hetero*)*, 144
sigma_poly *(*simulate_hetero*)*, 144
sigma_power *(*simulate_hetero*)*, 144
sigma_sin *(*simulate_hetero*)*, 144
sigma_spatial *(*simulate_hetero*)*, 144
sigma_step *(*simulate_hetero*)*, 144
sigma_u_shape *(*simulate_hetero*)*, 144
simulate_arch1 *(*simulate_hetero*)*, 144
simulate_hetero, 144

simulate_power, *145*
simulate_power *(*simulation_framework*)*, 146
simulate_power_analysis *(*simulation_framework*)*, 146
simulate_size, *145*
simulate_size *(*simulation_framework*)*, 146
simulate_type_I_errors *(*simulation_framework*)*, 146
simulation_framework, 146
stats::glm, *58*, *98*
stats::glm(), *165*
stats::integrate(), *59*
stats::lm, *43*, *45*, *58*, *66*, *71*, *82*, *84*, *88*, *92*, *94*, *98*, *107*, *111*, *113*, *115*, *130*, *142*
stats::lm(), *22*, *165*
stats::nobs(), *152*
std_error, 147
std_warning, 148
suggestDiagnosticsForProfile *(*generateHeteroRecommendations*)*, 27
suggestRemediation, 148
summary.hetero_grouped_suite *(*summary.hetero_test_suite*)*, 150
summary.hetero_test_suite, *30*, *135*, 150
summary.HeteroDiagnostic, 149
summary.lm(), *150*, *152*

t.test, *163*
test *(*test.HeteroDiagnostic*)*, 154
test.HeteroDiagnostic, 154
test_factory, 156
test_results, *11, 12*, *16*, *27*, *34, 35*, *38–40*, *42–47*, *49–53*, *55–58*, *61–63*, *65, 66*, *68*, *70–74*, *76–84*, *86–94*, *97*, *100*, *102*, *106, 107*, *109*, *111–116*, *118*, 156, *164*
TestFactory, 154, *156*
theme_hetero, 163
tidy.hetero_grouped_suite *(*tidy.hetero_test*)*, 164
tidy.hetero_test, 164
tidy.hetero_test_suite *(*tidy.hetero_test*)*, 164

validateTestInputs, 165
validateTestInputs(), *17*
var.test(), *151*

warnDiagnosticAssumptions *(*generateHeteroRecommendations*)*, 27