

Package: heteroTests (via r-universe)

October 12, 2026

Title Heteroscedasticity Diagnostics for Linear Models

Version 0.12.0

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>](https://doi.org/10.2307/1912934), Breusch and Pagan (1979) [<doi:10.2307/1911963>](https://doi.org/10.2307/1911963), Koenker (1981) [<doi:10.1016/0304-4076\(81\)90062-2>](https://doi.org/10.1016/0304-4076(81)90062-2), Goldfeld and Quandt (1965) [<doi:10.1080/01621459.1965.10480811>](https://doi.org/10.1080/01621459.1965.10480811) and Harvey (1976) [<doi:10.2307/1913974>](https://doi.org/10.2307/1913974); the score test of Cook and Weisberg (1983) [<doi:10.1093/biomet/70.1.1>](https://doi.org/10.1093/biomet/70.1.1); the ARCH test of Engle (1982) [<doi:10.2307/1912773>](https://doi.org/10.2307/1912773); and group-wise tests of equal variance, including those of Bartlett (1937) [<doi:10.1098/rspa.1937.0109>](https://doi.org/10.1098/rspa.1937.0109), Brown and Forsythe (1974) [<doi:10.1080/01621459.1974.10482955>](https://doi.org/10.1080/01621459.1974.10482955) and Hartley (1950) [<doi:10.2307/2332383>](https://doi.org/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

Roxygen list(markdown = 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, 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>

Config/pak/sysreqs libssl-dev

Repository <https://diogoribeiro7.r-universe.dev>

Date/Publication 2026-10-12 21:44:07 UTC

RemoteUrl <https://github.com/diogoribeiro7/heterotests>

RemoteRef HEAD

RemoteSha d2c02cfb3e18cc086d1f132e2e42ba6d6e0ea1b9

Contents

.diagnostic_registry	4
additional_tests	5
analyzeMLResiduals	5
autoCompareRemediations	6
autoplot.hetero_test_suite	6
autoTransform	7
bootstrap_methods	8
cachedTest	8
checkData	9
checkModel	9
checkModelEnhanced	10
clearAnalysisCache	11
clearTestCache	11
compareModelDiagnostics	12
compareTestResults	13
diagnostic_data	13
downloadTeachingData	14
fitRobust	15
fitWLS	15
generateDiagnosticReport	17
generateHeteroRecommendations	18
hetero_data	20
hetero_test_suite	21
HeteroDiagnostic	21
ht_set_log_level	22
launchDiagnosticDashboard	22
listDiagnostics	23
modern_diagnostics	24
performArchLMTest	24
performBartlettTest	25

performBoxMTest	26
performBPRandomEffectsTest	27
performBPTest	28
performBPTestRobust	29
performBPTestStreaming	30
performBrownForsytheTest	31
performCookWeisbergTest	32
performDavidianCarrollTest	33
performFlignerKilleenTest	34
performGlejserTest	35
performGQTest	37
performHarrisonMcCabeTest	38
performHartleyFmaxTest	41
performHarveyTest	42
performHighDimensionalTest	43
performInfluenceDiagnostics	45
performKoenkerTest	45
performKoenkerTestStreaming	46
performLeveneTest	47
performMcLeodLiTest	48
performNCVTest	49
performOBrienTest	50
performParkTest	51
performPesaranTest	52
performQuantileRegressionTest	53
performRankPermutationTest	54
performRESETTest	55
performScatterDiagnostic	56
performSpatialHeteroTest	57
performSpearmanTest	58
performSpreadLevelTest	59
performStudentizedBPTest	59
performSztroeterTest	61
performVarianceFormTest	63
performVIFDiagnostic	66
performWhiteTest	67
performWhiteTestBootstrap	68
performWhiteTestRobust	70
performWhiteTestStreaming	71
performWildBootstrapTest	73
plot.HeteroDiagnostic	74
plot.power_analysis	75
plotBeforeAfter	75
plotBubbleVariance	76
plotDiagnosticSuite	77
plotDiagnosticSuiteEnhanced	77
plotResidualDensity	78
plotResidualQQ	78

plotResidualsFitted	79
plotResidualsFittedEnhanced	80
plotSpreadLevel	80
print.htest_enhanced	81
rbootstrap_test_statistic	81
rcalculateEffectSize	84
registerDiagnostic	85
registerPlot	85
restimate_test_power	86
robust_implementations	87
rrunAdvancedDiagnostics	87
run_benchmark_suite	88
runDiagnosticPlots	90
runDiagnostics	90
runHeteroTests	91
runHeteroTestsParallel	93
runMultivariateTests	94
runPanelTests	94
runSurveyHeteroTests	95
runTimeSeriesTests	96
rvalidate_against_reference	97
scale_colour_hetero_diagnostic	98
simulate_hetero	98
simulation_framework	100
std_error	101
std_warning	102
suggestRemediation	102
summary.HeteroDiagnostic	103
test.HeteroDiagnostic	104
test_factory	104
TestFactory	104
theme_hetero	106
tidy.hetero_test	106
validateTestInputs	107

Index	109
--------------	------------

.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	<i>Additional heteroscedasticity tests</i>
------------------	--

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	<i>Machine-learning residual analysis</i>
--------------------	---

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.
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 <code>lm</code> .
-------	---

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)
```

bootstrap_methods	<i>Bootstrap utilities for heteroscedasticity diagnostics</i>
-------------------	---

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	<i>Cache results of a heteroscedasticity test</i>
------------	---

Description

Avoids recomputing expensive diagnostics by hashing their inputs and reusing stored results.

Usage

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

Arguments

<code>test_name</code>	Character scalar naming the diagnostic registered in <code>.diagnostic_registry</code> or <code>.test_factory</code> .
<code>model</code>	Fitted model object supplied to the diagnostic.
<code>data</code>	Data frame used to fit the model.
<code>...</code>	Additional arguments forwarded to the diagnostic.
<code>use_cache</code>	Logical scalar indicating whether cached results may be used (defaults to <code>TRUE</code>).

Details

The cache key is created by hashing the test name, model coefficients, residuals, data, and additional arguments. When the key already exists in the cache environment the stored result is returned immediately; otherwise the diagnostic is executed and its output stored for future calls. Set `use_cache = FALSE` to force recomputation when the underlying model has changed.

Value

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

See Also

[clearTestCache\(\)](#) resets the cache between analysis sessions.

checkData	<i>Validate data argument</i>
-----------	-------------------------------

Description

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

Usage

```
checkData(data)
```

Arguments

data	Object to check.
------	------------------

Value

Invisible data if valid, otherwise an error is thrown.

checkModel	<i>Validate model argument</i>
------------	--------------------------------

Description

Checks that model is of class lm or glm.

Usage

```
checkModel(model)
```

Arguments

model	An object to check.
-------	---------------------

Value

Invisible model if valid, otherwise an error is thrown.

checkModelEnhanced	<i>Enhanced model diagnostics</i>
--------------------	-----------------------------------

Description

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.

Usage

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

Arguments

<code>model</code>	Fitted model supplied to subsequent diagnostics.
<code>data</code>	Optional <code>data.frame</code> used when fitting <code>model</code> . When provided it enables multicollinearity checks via <code>performVIFDiagnostic()</code> .

Details

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)
checkModelEnhanced(fit)
```

clearAnalysisCache	<i>Clear cached diagnostic run results</i>
--------------------	--

Description

Removes memoised outputs stored by [runHeteroTests](#) when called with `use_cache = TRUE`. Useful for deterministic test runs or after mutating the underlying data/model so cached summaries should be invalidated.

Usage

```
clearAnalysisCache()
```

See Also

[runHeteroTests](#), [clearTestCache](#)

clearTestCache	<i>Clear cached test results</i>
----------------	----------------------------------

Description

Removes all objects from the internal cache environment so that subsequent diagnostic calls recompute their statistics.

Usage

```
clearTestCache()
```

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", "breusch_pagan"))
```

Arguments

<code>models</code>	List of fitted models or formulas.
<code>data</code>	Optional data frame when formulas are supplied.
<code>tests</code>	Diagnostics to run.

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	<i>Summarise heteroscedasticity test results</i>
--------------------	--

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", "breusch_pagan"))
```

Arguments

model	A fitted lm model or a formula.
data	Data frame used if model is a formula.
tests	Character vector of heteroscedasticity tests.

Value

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

Examples

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

diagnostic_data	<i>Simulated dataset with collinearity and mild nonlinearity</i>
-----------------	--

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

<code>name</code>	Name of the dataset to download. Only "tips" is currently supported.
<code>destfile</code>	Optional path to save the downloaded dataset. Defaults to a temporary file.
<code>quiet</code>	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	<i>Robust regression wrapper</i>
-----------	----------------------------------

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.

Value

An object of class rlm.

Examples

```
data(mtcars)
m <- fitRobust(mpg ~ wt + qsec, mtcars)
summary(m)
```

fitWLS	<i>Weighted Least Squares wrapper</i>
--------	---------------------------------------

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 <code>lm</code> .
data	Optional <code>base::data.frame</code> holding the variables named in <code>var_formula</code> . It is needed only when they are not in the model frame, and is matched to the fitted rows by row name.
var_formula	One-sided formula naming the variance regressors, for example <code>~ x</code> . <code>NULL</code> , the default, uses the regressors of <code>model</code> .
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 σ_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, and the fitted variance function (its form, regressors and coefficients) as the "variance_function" attribute, which is what `performVarianceFormTest()` reads.

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()`.

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.

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

<code>data</code>	A data.frame containing the modelling variables.
<code>response</code>	Optional response variable name used for labelling.
<code>profile</code>	Dataset profile produced by <code>analyseDatasetCharacteristics()</code> .
<code>test_results</code>	Named list of <code>htest</code> objects, typically returned by <code>runHeteroTests()</code> .
<code>alpha</code>	Significance level applied when interpreting p-values.
<code>model</code>	Fitted <code>lm/glm</code> object or model formula.
<code>interpretations</code>	Output from <code>interpretHeteroTestResults()</code> .
<code>recommendations</code>	Output from <code>suggestDiagnosticsForProfile()</code> , used to build the decision tree.
<code>remediation</code>	Output from <code>recommendRemediationStrategies()</code> .
<code>assumption_warnings</code>	Character vector of generated warnings.
<code>decision_tree</code>	Data frame returned by <code>buildDiagnosticDecisionTree()</code> .
<code>include_report</code>	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)
```

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	<i>S3 containers for heteroscedasticity diagnostics</i>
-------------------	---

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 `generics::tidy()`, `generics::glance()`, `generics::augment()` and `ggplot2::autoplot()`.

HeteroDiagnostic	<i>Create a diagnostic object</i>
------------------	-----------------------------------

Description

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

Usage

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

Arguments

<code>model</code>	A fitted <code>lm</code> or <code>glm</code> object, or a formula.
<code>data</code>	Data used to fit the model when <code>model</code> is a formula.

Value

An object of class `HeteroDiagnostic`.

Examples

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

ht_set_log_level	<i>Control logging for heteroTests diagnostics</i>
------------------	--

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.

launchDiagnosticDashboard	<i>Interactive diagnostic dashboard</i>
---------------------------	---

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	<i>List the registered diagnostics</i>
-----------------	--

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	<i>Modern heteroscedasticity diagnostics</i>
--------------------	--

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	<i>Perform Engle's ARCH LM test</i>
-------------------	-------------------------------------

Description

Regresses squared residuals on their lags to detect ARCH effects.

Usage

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

Arguments

<code>model</code>	an object of class <code>lm</code> .
<code>lags</code>	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.

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)
performArchLMTTest(m, lags = 2)
```

performBartlettTest	<i>Perform Bartlett's test for equality of variances</i>
---------------------	--

Description

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

Usage

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

Arguments

<code>model</code>	an object of class <code>lm</code> .
<code>data</code>	data frame used to fit <code>model</code> .
<code>group</code>	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.

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](https://doi.org/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](https://doi.org/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.

Usage

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

Arguments

data	A numeric data frame or matrix.
group	A factor indicating group membership.

Value

An object of class htest.

Examples

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

Arguments

model	an object of class <code>lm</code> .
data	data frame used to fit model.
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`.

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)
performBreuschPaganTest(model, data)
```

Arguments

model	an object of class <code>lm</code> .
data	data frame used to fit model.

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](#) or [performStudentizedBPTest](#).

Value

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

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\(\)](#).

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](#)

Examples

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

performBPTestRobust	<i>Robust Breusch–Pagan test with bootstrap and effect sizes</i>
---------------------	--

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,
  studentized = TRUE,
  bootstrap = FALSE,
  B = 1000,
  ci_level = 0.95,
  parallel = FALSE
)
```

Arguments

model	A fitted stats::lm object describing the mean structure whose residual variance is to be assessed.
data	A base::data.frame (or compatible object) containing the variables referenced in model. The data must include all observations used to fit model and should not contain unresolved missing values.
studentized	Logical, use studentized residuals for the auxiliary regression (Koenker variant)? Defaults to TRUE.
bootstrap	Logical, compute bootstrap diagnostics for the statistic and p-value?
B	Number of bootstrap replications when bootstrap = TRUE.
ci_level	Confidence level for reported intervals.
parallel	Logical, allow parallel bootstrap evaluation when the parallel package is available.

Details

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

Value

An augmented `htest` object containing a `robust_details` element describing the additional diagnostics.

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 [performStudentizedBPTest\(\)](#) 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 using chunked cross-products 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, chunk_size = 10000, progress = interactive())
```

Arguments

<code>model</code>	A fitted stats::lm object describing the mean structure whose residual variance is to be assessed.
<code>data</code>	A base::data.frame containing the variables referenced in <code>model</code> . The data must include all observations used to fit the model.
<code>chunk_size</code>	Positive integer giving the number of observations processed per streaming chunk. Smaller values reduce peak memory usage at the expense of additional iteration overhead.
<code>progress</code>	Logical flag controlling whether a textual progress bar is displayed while chunks are processed. Defaults to <code>interactive()</code> .

Details

Each chunk contributes to the cross-product matrices $X'X$ and $X'y$ for the auxiliary regression of squared residuals on the original regressors. The chunks are aggregated to recover the exact Breusch-Pagan statistic while keeping memory usage bounded. When **Matrix** is installed, sparse cross-products are used automatically for large chunks.

Value

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

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, group)
```

Arguments

model	an object of class <code>lm</code> .
data	data frame used to fit model.
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.

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.

Usage

```
performCookWeisbergTest(model)
```

Arguments

`model` an object of class `lm`.

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.

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](#), [performBPTTest](#)

Examples

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

```
performDavidianCarrollTest
```

Perform Davidian-Carroll test

Description

Regresses $\log(\text{residual}^2)$ on fitted values using a polynomial.

Usage

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

Arguments

model	an object of class <code>lm</code> .
degree	polynomial degree.

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`.

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

Non-parametric test based on ranks.

Usage

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

Arguments

model	an object of class <code>lm</code> .
data	data frame used to fit model.
group	name of the grouping variable.

Details

Absolute residuals are ranked after adjusting for group medians. The statistic 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.

References

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

Examples

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

performGlejserTest	<i>Perform Glejser test for heteroscedasticity</i>
--------------------	--

Description

Regresses absolute residuals on a transformation of a suspected variable. With `robust = TRUE` the test is the version of Im (2000) and of Machado and Santos Silva (2000), which keeps its level when the errors are asymmetric.

Usage

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

Arguments

<code>model</code>	an object of class <code>lm</code> .
<code>data</code>	data frame used to fit <code>model</code> .
<code>variable</code>	name of the suspected variable.
<code>transformation</code>	transformation applied to <code>variable</code> .
<code>robust</code>	logical. When <code>FALSE</code> (the default) 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. When <code>TRUE</code> the absolute residuals are corrected for the asymmetry of the errors before the auxiliary regression. See the section on asymmetric errors.

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.

The default statistic has its nominal level when positive and negative errors are equally likely. Use `robust = TRUE` when the residuals are 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 are recorded in `inst/validation/pass-a-size-power.csv` for the default and in `inst/validation/glejser-skewed-error` for both.

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,

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

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` the default statistic 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

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

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 the default. The two p-values agree as the sample grows.

In the same study the corrected statistic is within simulation error of the 5% level from 150 observations on. At 50 observations it rejects 5.7% to 6.6% of the time under skewed errors. Under symmetric errors $\hat{m}$ is close to zero and the two statistics differ little: the corrected one rejects about half a percentage point 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.

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)
performGlejserTest(m, mtcars, "wt")

# The version that keeps its level under asymmetric errors
performGlejserTest(m, mtcars, "wt", robust = TRUE)

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

performGQTest	<i>Perform Goldfeld-Quandt test for heteroscedasticity</i>
---------------	--

Description

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

Usage

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

Arguments

model	an object of class <code>lm</code> .
data	data frame used to fit model.
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 original model is re-estimated on the lower and upper segments. 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)`.

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.

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](https://doi.org/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,
  order_by,
  point = 0.5,
  alternative = c("less", "two.sided", "greater")
)
```

Arguments

model	A fitted <code>stats::lm</code> object, with or without weights.
data	A <code>base::data.frame</code> containing the variables used to fit model and the ordering variable named by order_by.
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 <code>floor(point * n)</code> observations, as in <code>lmtest::hmctest()</code> . 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 <code>performGQTest()</code> and in the default of <code>skedastic::harrison_mccabe()</code> .

Details

With $\hat{e}_{(1)}, \dots, \hat{e}_{(n)}$ 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,

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

where the z_j are independent standard normal and the λ_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 $(m - \sum_{i \leq m} h_{(i)}) / (n - k)$, 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 `hctest` 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.

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.

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::hmcetest()` 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.

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, group)
```

Arguments

model	A fitted lm object.
data	Data frame used to fit model.
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.

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	<i>Perform Harvey test for multiplicative heteroscedasticity</i>
-------------------	--

Description

Harvey's (1976) Lagrange multiplier test, which regresses $\log(\text{residuals}^2)$ on a set of variance regressors.

Usage

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

Arguments

model	an object of class <code>lm</code> .
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 <code>FALSE</code> (the default) the chi-square statistic $\text{ESS}/(\pi^2/2)$ 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 $\text{ESS}/(\pi^2/2)$, 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 `hstest` 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’.

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 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,
  variance_threshold = 0.9,
  max_components = NULL
)
```

Arguments

model	A fitted <code>stats::lm</code> object describing the mean structure whose residual variance is to be assessed.
data	A <code>base::data.frame</code> (or compatible object) containing the variables referenced in model. The data must include all observations used to fit model and should not contain unresolved missing values.
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.

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)
```

Arguments

model	an object of class lm.
data	data frame used to fit model.

Details

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

Value

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

Weighted fits

`lm(..., weights = w)` states that the error variance is σ^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.

References

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

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) using streamed cross-products so that large datasets can be processed without allocating the full auxiliary regression matrix.

Usage

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

Arguments

model	A fitted stats::lm object supplying residuals and fitted values for the diagnostic.
data	A base::data.frame containing the variables referenced in model. The data must align with the observations used to fit the model.
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 <code>interactive()</code> .

Details

Chunks of the data contribute to the cross-product matrices for the regression of squared residuals on the model regressors. Aggregating these matrices yields the exact Koenker statistic while bounding memory usage. Sparse cross-products from **Matrix** are used automatically when available for large chunks.

Value

A `htest` object equivalent to [performKoenkerTest](#) with metadata recording the streaming configuration.

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	<i>Perform Levene's test for equality of variances</i>
-------------------	--

Description

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

Usage

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

Arguments

model	an object of class <code>lm</code> .
data	data frame used to fit model.
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.

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	<i>Perform McLeod-Li test</i>
---------------------	-------------------------------

Description

Applies the Ljung-Box test to squared residuals.

Usage

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

Arguments

<code>model</code>	an object of class <code>lm</code> .
<code>lags</code>	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.

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	<i>Perform the non-constant variance (NCV) score test</i>
----------------	---

Description

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

Usage

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

Arguments

<code>model</code>	an object of class <code>lm</code> .
<code>var_formula</code>	optional one-sided formula giving the variance model, evaluated in the model frame of <code>model</code> (for example <code>~ x1 + x2</code>). When <code>NULL</code> the fitted values are used.

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 `hstest` containing the chi-square score statistic, its degrees of freedom and the p-value.

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.

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)
performNCVTest(m)

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

performOBrienTest	<i>Perform O’Brien test for equality of variances</i>
-------------------	---

Description

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

Usage

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

Arguments

model	A fitted lm object.
data	Data frame used to fit model and containing the grouping variable.
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.

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	<i>Perform Park test for heteroscedasticity</i>
-----------------	---

Description

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

Usage

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

Arguments

<code>model</code>	an object of class <code>lm</code> .
<code>data</code>	data frame used to fit <code>model</code> .
<code>variable</code>	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.

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	<i>Perform Pesaran CD test</i>
--------------------	--------------------------------

Description

Checks for cross-sectional dependence in panel residuals.

Usage

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

Arguments

<code>model</code>	an object of class <code>lm</code> .
<code>data</code>	data frame used to fit <code>model</code> .
<code>id</code>	individual identifier column.
<code>time</code>	time column.

Details

The statistic averages pairwise correlation coefficients of the residuals and is standardized to follow a normal distribution under the null of independence.

Value

An object of class `htest`.

References

Pesaran, M. H. (2004). General diagnostic tests for cross section dependence in panels. *Cambridge Working Papers in Economics*.

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,
  taus = c(0.25, 0.75),
  se_type = c("nid", "ker"),
  iid = TRUE
)
```

Arguments

<code>model</code>	A fitted <code>stats::lm</code> object describing the mean structure whose conditional quantile slopes are to be compared.
<code>data</code>	A <code>base::data.frame</code> containing the variables referenced in <code>model</code> .
<code>taus</code>	Numeric vector containing at least two distinct quantiles strictly between zero and one.
<code>se_type</code>	Standard-error method used by <code>quantreg::anova.rqs()</code> ; supported values are "nid" and "ker".
<code>iid</code>	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 σ^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 `hstest` object containing the F-like joint statistic, numerator and denominator degrees of freedom, p-value, fitted quantiles, and quantile-specific slope estimates.

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,
  order_by = NULL,
  B = 999,
  progress = interactive()
)
```

Arguments

model	A fitted <code>stats::lm</code> object describing the mean structure whose residual variance is to be assessed.
data	A <code>base::data.frame</code> (or compatible object) containing the variables referenced in model. The data must include all observations used to fit model and should not contain unresolved missing values.
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.
progress	Logical toggle for progress reporting during permutations.

Value

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

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	<i>Ramsey's RESET test for nonlinearity</i>
------------------	---

Description

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

Usage

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

Arguments

model	A fitted <code>lm</code> model.
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.

Value

An object of class htest.

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	data frame used to fit model.
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.

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,
  listw,
  permutations = 499,
  zero.policy = NULL
)
```

Arguments

model	A fitted <code>stats::lm</code> object describing the mean structure whose residual variance is to be assessed.
data	A <code>base::data.frame</code> (or compatible object) containing the variables referenced in model. The data must include all observations used to fit model and should not contain unresolved missing values.
listw	Spatial weights in <code>spdep listw</code> format or a numeric matrix coercible via <code>spdep::mat2listw()</code> .
permutations	Number of Monte Carlo permutations used to compute the reference distribution.
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.

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	<i>Perform Spearman rank correlation test</i>
---------------------	---

Description

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

Usage

```
performSpearmanTest(model)
```

Arguments

model an object of class lm.

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.

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 $\log(|\text{residuals}|)$ on $\log(\text{fitted})$.

Usage

```
performSpreadLevelTest(model)
```

Arguments

`model` an object of class `lm`.

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`.

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*

Description

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.

Usage

```
performStudentizedBPTest(model, data)
```

Arguments

model	A fitted <code>stats::lm</code> object providing the residuals and design matrix for the auxiliary regression.
data	A <code>base::data.frame</code> containing the variables used to fit model. It must include all observations referenced by the model object.

Details

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 implementation shares the validation helpers used across the package to ensure that: (i) the model and data satisfy minimum sample-size thresholds via `rvalidateModelInputs()` and `rvalidateDataInputs()`, (ii) missing values are handled by `rhandleMissingValues()`, 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.

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)
performStudentizedBPTest(mod, mtcars)

# Detect heteroscedasticity driven by a single regressor
set.seed(321)
x <- runif(180)
y <- 5 - 1.5 * x + rnorm(180, sd = 0.4 + 0.6 * x)
df <- data.frame(y, x)
performStudentizedBPTest(lm(y ~ x, data = df), df)
```

performSzoeterTest	<i>Szoeter test for ordered alternatives</i>
--------------------	--

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 Szoeter (1978).

Usage

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

Arguments

model	A fitted <code>stats::lm</code> object representing the mean equation under study.
data	A <code>base::data.frame</code> (or compatible tibble) containing the variables used to fit model and the ordering variable supplied via <code>order_by</code> .
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 <code>order_by</code> , "less" for variance decreasing, and "two.sided" for a change in either direction. Szoeter'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 $h_i = i$. Under homoskedasticity h is centred on the mean rank $(n + 1)/2$ with variance $(n^2 - 1)/(6n)$, so

$$Q = \frac{h - (n + 1)/2}{\sqrt{(n^2 - 1)/(6n)}}$$

is asymptotically standard normal. Variance that grows with `order_by` shifts weight onto the high ranks and drives Q upwards.

The implementation relies on the package validation helpers to align the supplied data with the model residuals, 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.

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 asymptotic normal reference distribution.

Validation

The statistic and its null variance are checked against an independent reconstruction of Szroeter (1978) in `'tests/testthat/test-pass-a-reference.R'`. 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'`.

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.

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)
performSztroeterTest(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)
performSztroeterTest(lm(y ~ x, data = df), df, order_by = "x")

# Two-sided version when the direction of the change is not known in advance
performSztroeterTest(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 <code>stats::lm</code> object for the mean equation, or the result of <code>fitWLS()</code> . In the second case the variance function that was fitted is the one tested, and <code>var_formula</code> and <code>form</code> must be left at their defaults.
data	Optional <code>base::data.frame</code> holding the variables named in <code>var_formula</code> or <code>against</code> . It is needed only when they are not in the model frame, and is matched to the fitted rows by row name.

var_formula	One-sided formula naming the variance regressors z_i , for example $\sim x$. NULL, the default, uses the regressors of <code>model</code> .
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 <code>fitWLS()</code> .
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 $\sim w$ or $\sim I(x^3)$ 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 $r_i = \hat{e}_i^2 / \hat{\sigma}_i^2$ 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 γ 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`.

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	<i>Variance inflation factors</i>
----------------------	-----------------------------------

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	<i>Perform White's test for heteroscedasticity</i>
------------------	--

Description

Implements White's test on a fitted linear model.

Usage

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

Arguments

model	an object of class <code>lm</code> .
data	Data frame used to fit model.
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 with degrees of freedom equal to the number of regressors in the auxiliary model.

Value

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

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\(\)](#).

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](https://doi.org/10.2307/1912934)

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

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, B = 1000, parallel = FALSE)
```

Arguments

model	A fitted <code>stats::lm</code> object representing the mean specification to be diagnosed.
data	A <code>base::data.frame</code> (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.
B	Integer scalar giving the number of bootstrap replications. Larger values yield smoother p-value estimates at the cost of additional runtime.
parallel	Logical scalar; if TRUE the bootstrap loop is executed with <code>parallel::mclapply()</code> when the parallel package and forked processing are available.

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_{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 re-sampling 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.

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 with an offset the model was given as an argument, `lm(..., offset = o)`; an offset written in the formula, `offset(o)`, is evaluated again from the data of the replicate.

The weights are held fixed in every replicate. When they were estimated from the same data, as by `fitWLS()`, the replicates leave that estimation out. In `inst/validation/weighted-fits-size.R`, on data whose standard deviation is `exp(0.3x)`, with the variance function correctly specified, normal errors, 150 observations and $B = 499$, the rejection rate at the 5% level was 0.064. In the same design with the true weights it was 0.054, and with the weights estimated again in each replicate, which this function does not do, 0.042. The section on weighted fits in `performWildBootstrapTest()` has the larger error of Koenker's statistic.

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 forks parallel::detectCores() - 1 workers, which Windows
# does not support. The call runs only where that is at most two workers,
# the most R CMD check --as-cran allows.
if (.Platform$OS.type != "windows" && isTRUE(parallel::detectCores() <= 3L)) {
```

```
performWhiteTestBootstrap(mod, sim, B = 199, parallel = TRUE)
}
```

```
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,
  method = c("standard", "reduced"),
  bootstrap = FALSE,
  B = 1000,
  ci_level = 0.95,
  parallel = FALSE
)
```

Arguments

<code>model</code>	A fitted <code>stats::lm</code> object representing the mean specification to be diagnosed.
<code>data</code>	A <code>base::data.frame</code> (or object coercible to one) containing the variables referenced by <code>model</code> . It must include the observations used to fit <code>model</code> and will be checked for missing values.
<code>method</code>	Character string selecting the auxiliary specification. "standard" retains squares and cross-products, while "reduced" excludes cross-products for high-dimensional designs.
<code>bootstrap</code>	Logical, compute bootstrap diagnostics for the statistic and p-value?
<code>B</code>	Number of bootstrap replications when <code>bootstrap = TRUE</code> .
<code>ci_level</code>	Confidence level for reported intervals.
<code>parallel</code>	Logical, allow parallel bootstrap evaluation when the <code>parallel</code> package is available.

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.

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 via chunked cross-products 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,
  chunk_size = 10000,
  cross_products = TRUE,
  max_interactions = 10,
  progress = interactive()
)
```

Arguments

<code>model</code>	A fitted <code>stats::lm</code> object representing the mean specification to be diagnosed.
<code>data</code>	A <code>base::data.frame</code> (or object coercible to one) containing the variables referenced by <code>model</code> . It must include the observations used to fit <code>model</code> and will be checked for missing values.
<code>chunk_size</code>	Positive integer specifying the number of observations per chunk. Smaller values reduce memory usage at the expense of additional iteration overhead.
<code>cross_products</code>	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.
<code>max_interactions</code>	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.
<code>progress</code>	Logical flag indicating whether a progress bar should be displayed while streaming the data. Defaults to <code>interactive()</code> .

Details

The streaming implementation iteratively builds the cross-product matrices required for the auxiliary regression without materialising the full design matrix. Each chunk contributes to $X'X$, $X'y$, and summary statistics for the response. The final nR^2 statistic is then computed exactly as in the standard White test, ensuring numerical equivalence while dramatically reducing peak memory usage. When `Matrix` is installed, sparse cross-products are leveraged automatically for large chunks.

Value

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

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 χ^2 approximation.

Usage

```
performWildBootstrapTest(
  model,
  data,
  B = 499,
  distribution = c("rademacher", "mammen"),
  progress = interactive()
)
```

Arguments

model	A fitted <code>stats::lm</code> object describing the mean structure whose residual variance is to be assessed.
data	A <code>base::data.frame</code> (or compatible object) containing the variables referenced in model. The data must include all observations used to fit model and should not contain unresolved missing values.
B	Integer number of bootstrap replications. Defaults to 499.
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.

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 $(1 + \#\{T_b^* \geq T\}) / (B + 1)$. Under H_0 this controls size even for heavy-tailed errors; 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.

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.

The weights are held fixed in every replicate. When they were estimated from the same data, as by `fitWLS()`, the replicates leave that estimation out, and the test inherits the error of the tests that take the weights as known. In `inst/validation/weighted-fits-size.R`, on data whose standard deviation is $\exp(0.3x)$, with the variance function correctly specified, normal errors and $B = 499$, its rejection rate at the 5% level was 0.086 with 150 observations and 0.105 with 600. In the same design with the true weights it was 0.054 and 0.048, and with the weights estimated again in each replicate, which this function does not do, 0.060 and 0.051. `performVarianceFormTest()` allows for the estimation.

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	<i>Plot method for power analysis results</i>
---------------------	---

Description

Plot method for power analysis results

Usage

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

Arguments

x Object returned by simulate_power_analysis().

... Further arguments passed to the underlying plotting routines.

plotBeforeAfter	<i>Compare residuals before and after remediation</i>
-----------------	---

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	<i>Bubble plot of residual variance by covariate</i>
--------------------	--

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	<i>Generate a suite of diagnostic plots</i>
---------------------	---

Description

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

Usage

```
plotDiagnosticSuite(model)
```

Arguments

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

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	<i>Enhanced diagnostic plot suite</i>
-----------------------------	---------------------------------------

Description

Returns a list of improved diagnostic plots with statistical overlays.

Usage

```
plotDiagnosticSuiteEnhanced(model)
```

Arguments

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

Value

A list of ggplot objects.

plotResidualDensity	<i>Density plot of residuals</i>
---------------------	----------------------------------

Description

Shows the distribution of residuals with a kernel density estimate.

Usage

```
plotResidualDensity(model)
```

Arguments

model	A fitted model of class <code>lm</code> or <code>glm</code> .
-------	---

Value

A ggplot object.

Examples

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

plotResidualQQ	<i>QQ plot of residuals</i>
----------------	-----------------------------

Description

Visualises departure from normality using a QQ plot.

Usage

```
plotResidualQQ(model)
```

Arguments

model	A fitted model of class <code>lm</code> or <code>glm</code> .
-------	---

Value

A ggplot object.

Examples

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

plotResidualsFitted	<i>Plot residuals vs fitted values</i>
---------------------	--

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)
```

Arguments

model	A fitted model of class <code>lm</code> .
-------	---

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.

Value

A ggplot object.

Examples

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

`plotResidualsFittedEnhanced`*Enhanced residuals vs fitted plot*

Description

Highlights influential observations and includes a LOESS smooth with confidence bands.

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()`.

```
rbootstrap_test_statistic
```

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

```
rbootstrap_test_statistic(
  test_function,
  model,
  data,
  B = 1000,
  parallel = FALSE,
  ci_level = 0.95,
  resample = c("null", "pairs"),
  n_cores = NULL,
  progress = interactive(),
  ...
)
```

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. A glm is rejected: each replicate is refitted with least squares, which would discard its family and link.
data	Data frame used when fitting model, holding the rows it was fitted to under their row names. Other rows are left out. See Details.
B	Integer, number of bootstrap replications. Defaults to 1000.
parallel	Logical, compute replicates in parallel when possible?
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.
progress	Logical flag controlling whether a textual progress bar is displayed during bootstrap replication.
...	Additional arguments forwarded to test_function.

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. The weights are held fixed; when they were estimated from the same data, see the section on weighted fits in [performWildBootstrapTest\(\)](#).

Weighted or not, each replicate is refitted with an offset the model was given as an argument, `lm(..., offset = o)`; an offset written in the formula, `offset(o)`, is evaluated again from the

data of the replicate. 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 model is then refitted to the rows of data, and data is refused when that refit does not give the residuals of the fit to a relative $1e-8$: rows in another order taken by position, values that differ from those the model was fitted to, or a variable the formula finds outside data, which keeps the order of the fit. Pairs resampling of a model with such a variable, for example `offset(o)` with `o` in the environment of the formula, is refused too; give such an offset as an argument instead.

Null-imposed resampling additionally requires the response to be a plain variable. For $\log(y) \sim x$ 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.

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.

Examples

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

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.

Examples

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

registerDiagnostic	<i>Register a custom diagnostic</i>
--------------------	-------------------------------------

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.

Value

Invisibly returns NULL.

See Also

[listDiagnostics](#) for the names already registered.

Examples

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

registerPlot	<i>Register a diagnostic plot</i>
--------------	-----------------------------------

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())
```

reestimate_test_power	<i>Estimate test power from observed effect size</i>
-----------------------	--

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

```
reestimate_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).

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.

rrunAdvancedDiagnostics

Run enhanced heteroscedasticity diagnostics

Description

Executes one or more robust diagnostics with automatic enhancements for small samples and optional studentisation.

Usage

```
rrunAdvancedDiagnostics(
  model,
  data,
  tests = "all",
  auto_enhance = TRUE,
  bootstrap_B = 500,
  parallel = FALSE,
  ci_level = 0.95
)
```

Arguments

model	A fitted <code>stats::lm</code> object representing the mean specification to be diagnosed.
data	A <code>base::data.frame</code> (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 <code>c("white", "bp")</code> .
auto_enhance	Logical, enable automatic bootstrap for small samples ($n < 50$) and studentization for Breusch-Pagan.
bootstrap_B	Number of bootstrap replications used when automatic enhancement triggers bootstrap.

parallel	Logical, allow parallel bootstrap evaluation when the parallel package is available.
ci_level	Confidence level for reported intervals.

Value

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

run_benchmark_suite	<i>Benchmark heteroTests against reference implementations</i>
---------------------	---

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)
}

```

runDiagnosticPlots	<i>Run registered diagnostic plots</i>
--------------------	--

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	<i>Run a suite of model diagnostics</i>
----------------	---

Description

Combines heteroscedasticity checks with tests for collinearity, nonlinearity and influential observations.

Usage

```
runDiagnostics(
  model,
  data = NULL,
  tests = c("white", "breusch_pagan"),
  power = 2:3,
  use_cache = TRUE,
  chunk_threshold_mb = 100,
  chunk_size = 10000,
  progress = interactive()
)
```

Arguments

<code>model</code>	A fitted <code>lm</code> model or a formula.
<code>data</code>	Optional data frame if <code>model</code> is a formula.
<code>tests</code>	Heteroscedasticity tests to run.
<code>power</code>	Powers for <code>performRESETTest</code> .
<code>use_cache</code>	Logical flag forwarded to runHeteroTests controlling whether cached diagnostic results may be reused.
<code>chunk_threshold_mb</code>	Numeric threshold passed to runHeteroTests for enabling streaming diagnostics on large datasets.
<code>chunk_size</code>	Integer chunk size supplied to runHeteroTests when streaming diagnostics.
<code>progress</code>	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)
```

<code>runHeteroTests</code>	<i>Run multiple heteroscedasticity diagnostics</i>
-----------------------------	--

Description

Convenience wrapper that executes several heteroscedasticity tests on a fitted linear model.

Usage

```
runHeteroTests(
  model,
  data = NULL,
  tests = c("white", "breusch_pagan"),
  use_cache = TRUE,
  chunk_threshold_mb = 100,
  chunk_size = 10000,
  progress = interactive()
)
```

Arguments

model	an object of class <code>lm</code> .
data	optional data frame used to fit model. If omitted, <code>model.frame(model)</code> is used. With data grouped by <code>dplyr::group_by()</code> 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.
tests	character vector of test names to run: any of the names <code>listDiagnostics()</code> returns, which include those added with <code>registerDiagnostic</code> .
use_cache	logical indicating whether cached results should be reused when available. Requires the digest package and defaults to <code>TRUE</code> .
chunk_threshold_mb	numeric threshold (in megabytes) above which streaming implementations are preferred for supported diagnostics.
chunk_size	integer number of observations processed per chunk when streaming diagnostics.
progress	logical flag controlling whether textual progress bars are displayed when running multiple diagnostics.

Details

By default runs `performWhiteTest` and `performBPTest`, but additional tests can be requested. Custom diagnostics registered via `registerDiagnostic` are also available.

Value

A named list of `htest` objects.

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.

Examples

```
data(mtcars)
m <- lm(mpg ~ wt + qsec, data = mtcars)
# Default: White and Breusch-Pagan
runHeteroTests(m, mtcars)
# Specify additional diagnostics
runHeteroTests(m, mtcars, tests = c("white", "koenker", "ncv"))
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 <code>.test_factory</code> 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 <code>parallel::detectCores() - 1</code> , with a lower bound of 1.

Details

When `n_cores` exceeds one and the `parallel` package is available, the helper evaluates the diagnostics via `parallel::parLapply()` after exporting the model and data to the worker environment. On systems without fork support the function gracefully reverts to sequential execution.

Value

Named list of test results in the order provided by `tests`.

See Also

`cachedTest()` for memoised execution of individual diagnostics.

runMultivariateTests	<i>Run multivariate heteroscedasticity tests</i>
----------------------	--

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	<i>Run panel-data heteroscedasticity tests</i>
---------------	--

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.
id	Column identifying individuals.
time	Column identifying time periods for the Pesaran test.
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", "breusch_pagan"), ...)
```

Arguments

<code>formula</code>	Model formula specifying the mean structure.
<code>design</code>	A <code>survey::svydesign()</code> object. Only its data are used.
<code>tests</code>	Diagnostic names passed on to <code>runHeteroTests</code> .
<code>...</code>	Additional arguments forwarded to <code>runHeteroTests</code> .

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`.

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	<i>Run time-series heteroscedasticity tests</i>
--------------------	---

Description

Convenience wrapper for Engle's ARCH LM and McLeod-Li tests.

Usage

```
runTimeSeriesTests(model, lags = 1, tests = c("arch_lm", "mcleod_li"))
```

Arguments

<code>model</code>	A fitted <code>lm</code> object.
<code>lags</code>	Number of lags for both tests.
<code>tests</code>	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)
```

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 <code>stats::lm</code> object representing the mean specification to be diagnosed.
data	A <code>base::data.frame</code> (or object coercible to one) containing the variables referenced by <code>model</code> . It must include the observations used to fit <code>model</code> 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.

```
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.

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.

Usage

```
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	<i>Generate standardized error message</i>
-----------	--

Description

Generate standardized error message

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	<i>Generate standardized warning message</i>
-------------	--

Description

Generate standardized warning message

Usage

```
std_warning(type, ...)
```

Arguments

type	Warning message type.
...	Named arguments for the template.

Value

Issues a warning with a formatted message.

suggestRemediation	<i>Suggest remediation actions for heteroscedasticity</i>
--------------------	---

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 <code>runHeteroTests()</code> .
x	Object of class <code>remediation_suggestions</code> .
...	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.

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", "breusch_pagan"), ...)
```

Arguments

<code>object</code>	A <code>HeteroDiagnostic</code> object.
<code>tests</code>	Character vector naming the diagnostics to summarise.
<code>...</code>	Further arguments passed to the individual tests.

Value

Named numeric vector of test statistics.

test.HeteroDiagnostic	<i>Run diagnostic tests</i>
-----------------------	-----------------------------

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.

test_factory	<i>Shared instance of the test factory</i>
--------------	--

Description

Users can register new tests via this object. Built-in tests are pre-registered at package load time.

TestFactory	<i>Heteroscedasticity Test Factory</i>
-------------	--

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()`.

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.

theme_hetero	<i>Heteroscedasticity diagnostic theme</i>
--------------	--

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	<i>broom methods for heteroscedasticity diagnostics</i>
------------------	---

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

<code>x</code>	Heteroscedasticity diagnostic result(s) returned by <code>runHeteroTests</code> .
<code>data</code>	Optional data frame used to augment diagnostics; defaults to the data embedded within the fitted model when available.
<code>...</code>	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)
}
```

<code>validateTestInputs</code>	<i>Validate model and data inputs before running diagnostics</i>
---------------------------------	--

Description

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.

Usage

```
validateTestInputs(model, data, test_name, min_obs = 10)
```

Arguments

<code>model</code>	A fitted model created by <code>stats::lm()</code> or <code>stats::glm()</code> .
<code>data</code>	A <code>data.frame</code> containing the variables used to fit <code>model</code> .
<code>test_name</code>	Character scalar naming the diagnostic that is about to run; included in error messages for clarity.
<code>min_obs</code>	Non-negative integer giving the minimum sample size accepted by the diagnostic. Defaults to 10.

Details

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()`, `checkModelEnhanced()`, `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, [24](#)
 - .diagnostic_registry, [4](#)
- additional_tests, [5](#)
- analyseDatasetCharacteristics
 - (generateHeteroRecommendations), [18](#)
- analyzeMLResiduals, [5](#), [12](#)
- augment.hetero_test(tidy.hetero_test), [106](#)
- augment.hetero_test_suite(tidy.hetero_test), [106](#)
- autoCompareRemediations, [6](#)
- autoplot.hetero_grouped_suite(
autoplot.hetero_test_suite), [6](#)
- autoplot.hetero_test_suite, [6](#)
- autoTransform, [7](#)
- base::data.frame, [16](#), [29](#), [30](#), [38](#), [44](#), [47](#), [53](#),
[55](#), [57](#), [60](#), [61](#), [63](#), [68](#), [70](#), [72](#), [73](#), [87](#),
[97](#)
- bootstrap_methods, [8](#)
- buildDiagnosticDecisionTree
 - (generateHeteroRecommendations), [18](#)
- cachedTest, [8](#)
- cachedTest(), [11](#), [93](#)
- checkData, [9](#)
- checkModel, [9](#)
- checkModel(), [10](#), [108](#)
- checkModelEnhanced, [10](#)
- checkModelEnhanced(), [108](#)
- clearAnalysisCache, [11](#)
- clearTestCache, [11](#), [11](#)
- clearTestCache(), [9](#)
- compareModelDiagnostics, [5](#), [12](#)
- compareTestResults, [13](#)
- composeDiagnosticNarrative
 - (generateHeteroRecommendations), [18](#)
- diagnostic_data, [13](#)
- downloadTeachingData, [14](#)
- fitRobust, [15](#)
- fitWLS, [15](#), [28](#), [46](#), [50](#), [67](#)
- fitWLS(), [63–65](#), [69](#), [74](#)
- generate_benchmark_report
 - (run_benchmark_suite), [88](#)
- generateDiagnosticReport, [17](#)
- generateHeteroRecommendations, [18](#)
- ggplot2::scale_colour_manual(), [98](#)
- ggplot2::theme, [106](#)
- glance.hetero_test(tidy.hetero_test),
[106](#)
- glance.hetero_test_suite(
tidy.hetero_test), [106](#)
- hetero_data, [20](#)
- hetero_grouped_suite, [7](#)
- hetero_grouped_suite(
hetero_test_suite), [21](#)
- hetero_test_suite, [7](#), [21](#)
- HeteroDiagnostic, [21](#)
- ht_clear_log_history(
ht_set_log_level), [22](#)
- ht_enable_log_capture(
ht_set_log_level), [22](#)
- ht_log_history(ht_set_log_level), [22](#)
- ht_set_log_level, [22](#)
- interpretHeteroTestResults
 - (generateHeteroRecommendations),
[18](#)
- launchDiagnosticDashboard, [22](#)
- listDiagnostics, [23](#), [85](#), [92](#)

lmtest::bptest(), 60
 lmtest::hmcptest(), 39, 40
 modern_diagnostics, 24
 parallel::mclapply(), 68
 parallel::parLapply(), 93
 performArchLMTest, 24
 performBartlettTest, 25
 performBoxMTest, 26
 performBPRandomEffectsTest, 27
 performBPTTest, 28, 30, 31, 33, 43, 50, 92
 performBPTTest(), 29, 30, 60
 performBPTTestRobust, 29
 performBPTTestRobust(), 60
 performBPTTestStreaming, 30
 performBreuschPaganTest
 (performBPTTest), 28
 performBrownForsytheTest, 31
 performCookWeisbergTest, 32, 50
 performDavidianCarrollTest, 33
 performFlignerKilleenTest, 34
 performGlejserTest, 35, 43
 performGlejserTest(), 40
 performGQTest, 37
 performGQTest(), 39, 40, 62
 performHarrisonMcCabeTest, 38
 performHartleyFmaxTest, 41
 performHarveyTest, 36, 42
 performHarveyTest(), 65
 performHighDimensionalTest, 43
 performInfluenceDiagnostics, 45
 performKoenkerTest, 28, 32, 33, 36, 45, 47,
 49, 50, 67
 performKoenkerTest(), 60, 62, 65
 performKoenkerTestStreaming, 46
 performLeveneTest, 47
 performMcLeodLiTest, 48
 performNCVTest, 32, 33, 49
 performOBrienTest, 50
 performParkTest, 36, 43, 51
 performParkTest(), 65
 performPesaranTest, 52
 performQuantileRegressionTest, 53
 performRankPermutationTest, 54
 performRESETTest, 55
 performScatterDiagnostic, 56
 performSpatialHeteroTest, 57
 performSpearmanTest, 58
 performSpreadLevelTest, 59
 performStudentizedBPTTest, 28, 59
 performStudentizedBPTTest(), 30
 performSztroeterTest, 61
 performSztroeterTest(), 40
 performVarianceFormTest, 46, 63
 performVarianceFormTest(), 16, 17, 74
 performVIFDiagnostic, 66
 performVIFDiagnostic(), 10
 performWhiteTest, 67, 92
 performWhiteTest(), 69–72
 performWhiteTestBootstrap, 68
 performWhiteTestBootstrap(), 71
 performWhiteTestRobust, 70
 performWhiteTestRobust(), 69, 72
 performWhiteTestStreaming, 71
 performWildBootstrapTest, 73
 performWildBootstrapTest(), 69, 82
 plot.HeteroDiagnostic, 74
 plot.power_analysis, 75
 plotBeforeAfter, 75
 plotBubbleVariance, 76
 plotDiagnosticSuite, 77
 plotDiagnosticSuiteEnhanced, 77
 plotResidualDensity, 78
 plotResidualQQ, 78
 plotResidualsFitted, 79
 plotResidualsFittedEnhanced, 80
 plotSpreadLevel, 80
 print.hetero_recommendation_report
 (generateHeteroRecommendations),
 18
 print.htest_enhanced, 81
 print.remediation_suggestions
 (suggestRemediation), 102
 rbootstrap_test_statistic, 81
 rcalculateEffectSize, 84
 recommendRemediationStrategies
 (generateHeteroRecommendations),
 18
 registerDiagnostic, 85
 registerDiagnostic(), 23, 24
 registerPlot, 85
 reestimate_test_power, 86
 rhandleMissingValues(), 60
 robust_implementations, 87
 rrunAdvancedDiagnostics, 87
 run_benchmark_suite, 88

runDiagnosticPlots, 90
 runDiagnostics, 90
 runHeteroTests, 11, 91, 91, 95, 96
 runHeteroTests(), 23, 24
 runHeteroTestsParallel, 93
 runMultivariateTests, 94
 runMultivariateTests(), 23
 runPanelTests, 94, 97
 runSurveyHeteroTests, 95
 runTimeSeriesTests, 95, 96
 rvalidate_against_reference, 97
 rvalidateDataInputs(), 60
 rvalidateModelInputs(), 60, 108
 rvalidateTestRequirements(), 60, 108

 scale_colour_hetero_diagnostic, 98
 scale_fill_hetero_diagnostic
 (scale_colour_hetero_diagnostic),
 98
 sigma_exp_decay(simulate_hetero), 98
 sigma_exponential(simulate_hetero), 98
 sigma_gaussian_peak(simulate_hetero),
 98
 sigma_group(simulate_hetero), 98
 sigma_inverse(simulate_hetero), 98
 sigma_linear(simulate_hetero), 98
 sigma_logistic(simulate_hetero), 98
 sigma_multiplicative(simulate_hetero),
 98
 sigma_piecewise(simulate_hetero), 98
 sigma_piecewise_linear
 (simulate_hetero), 98
 sigma_poly(simulate_hetero), 98
 sigma_power(simulate_hetero), 98
 sigma_sin(simulate_hetero), 98
 sigma_spatial(simulate_hetero), 98
 sigma_step(simulate_hetero), 98
 sigma_u_shape(simulate_hetero), 98
 simulate_arch1(simulate_hetero), 98
 simulate_hetero, 98
 simulate_power_analysis
 (simulation_framework), 100
 simulate_type_I_errors
 (simulation_framework), 100
 simulation_framework, 100
 stats::glm(), 108
 stats::integrate(), 39
 stats::lm, 29, 30, 38, 44, 47, 53, 55, 57, 60,
 61, 63, 68, 70, 72, 73, 87, 97

 stats::lm(), 108
 std_error, 101
 std_warning, 102
 suggestDiagnosticsForProfile
 (generateHeteroRecommendations),
 18
 suggestRemediation, 102
 summary.HeteroDiagnostic, 103

 test(test.HeteroDiagnostic), 104
 test.HeteroDiagnostic, 104
 test_factory, 104
 TestFactory, 104
 theme_hetero, 106
 tidy.hetero_grouped_suite
 (tidy.hetero_test), 106
 tidy.hetero_test, 106
 tidy.hetero_test_suite
 (tidy.hetero_test), 106

 validateTestInputs, 107
 validateTestInputs(), 10

 warnDiagnosticAssumptions
 (generateHeteroRecommendations),
 18