Package: hrt (via r-universe)

October 25, 2026

Type Package

Title Heteroskedasticity Robust Testing

Version 1.0.2

Date 2025-04-14

Maintainer David Preinerstorfer <david.preinerstorfer@wu.ac.at>

Description Functions for testing affine hypotheses on the regression coefficient vector in regression models with heteroskedastic errors: (i) a function for computing various test statistics (in particular using HC0-HC4 covariance estimators based on unrestricted or restricted residuals); (ii) a function for numerically approximating the size of a test based on such test statistics and a user-supplied critical value; and, most importantly, (iii) a function for determining size-controlling critical values for such test statistics and a user-supplied significance level (also incorporating a check of conditions under which such a size-controlling critical value exists). The three functions are based on results in Poetscher and Preinerstorfer (2021) ``Valid Heteroskedasticity Robust Testing" <doi:10.48550/arXiv.2104.12597>, which will appear as <doi:10.1017/S0266466623000269>.

License GPL-2

Imports methods, stats, Rcpp, CompQuadForm

LinkingTo Rcpp, RcppEigen

NeedsCompilation yes

Author David Preinerstorfer [aut, cre]

Repository https://cran.r-universe.dev

Date/Publication 2025-04-17 07:20:05 UTC

RemoteUrl https://github.com/cran/hrt

RemoteRef HEAD

RemoteSha 629735305f602f2e9bc52668a7bf6c51ab5b5cc2

Contents

hrt-package 2
critical.value 3
size 7
test.stat 11

Index **13**

`hrt-package` *Heteroskedasticity Robust Testing*

Description

The package **hrt** provides three functions in the context of testing affine restrictions on the regression coefficient vector in linear models with heteroskedastic (but independent) errors. The methods implemented in **hrt** are based on the article Pötscher and Preinerstorfer (2021). The package can be used to compute various heteroskedasticity robust test statistics; to numerically determine size-controlling critical values when the error vector is heteroskedastic and Gaussian (or, more generally, elliptically symmetric); and to compute the size of a test that is obtained from a heteroskedasticity robust test statistic and a user-supplied critical value.

Details

hrt provides three functions:

1. The function `test.stat` can be used to evaluate the test statistics T_{uc}, T_{Het} (with HC0-HC4 weights), $\tilde{T}_{uc}$, or $\tilde{T}_{Het}$ (with HC0R-HC4R weights), as defined in Pötscher and Preinerstorfer (2021).
2. The function `critical.value` provides an implementation of Algorithm 3 in Pötscher and Preinerstorfer (2021), based on the auxiliary algorithm A equal to Algorithm 1 (if $q = 1$) or Algorithm 2 (if $q > 1$) in the same reference. This function can be used to determine size-controlling critical values for the test statistics T_{uc}, T_{Het} (with HC0-HC4 weights), $\tilde{T}_{uc}$, or $\tilde{T}_{Het}$ (with HC0R-HC4R weights), whenever such critical values exist (which is checked numerically when the algorithm is applied).
3. The function `size` provides an implementation of Algorithm 1 or 2, respectively, in Pötscher and Preinerstorfer (2021), depending on whether $q = 1$ or $q > 1$. Given a user-supplied critical value, the respective algorithm can be used to determine the size of a test based on one of the test statistics T_{uc}, T_{Het} (with HC0-HC4 weights), $\tilde{T}_{uc}$, or $\tilde{T}_{Het}$ (with HC0R-HC4R weights).

   We refer the user to the description of the three functions below, and to Pötscher and Preinerstorfer (2021) for details concerning the framework, the test statistics, the algorithms, and the underlying theoretical results.

References

Pötscher, B. M. and Preinerstorfer, D. (2021). Valid Heteroskedasticity Robust Testing. <arXiv:2104.12597>

`critical.value` *Critical Values for Heteroskedasticity Robust Testing*

Description

This function provides an implementation of Algorithm 3 in Pötscher and Preinerstorfer (2021), based on Algorithm 1 (if $q = 1$) or Algorithm 2 (if $q > 1$) in the same reference as the auxiliary algorithm A. Which of the two algorithms is used is automatically determined as a function of q, the number of rows of `R`.

The user is referred to Pötscher and Preinerstorfer (2021) for definitions, a detailed description of the problems solved by the algorithms, and for a detailed description of the algorithms themselves.

Most of the input parameters to `critical.value` are actually used in the auxiliary Algorithm 1 or 2, respectively. Algorithm 1 is based on the function `davies` from the package **CompQuadForm**. The parameters `lim` and `acc` for `davies` can be supplemented by the user. Algorithms 1 and 2 are implemented using the function `constrOptim` from **stats** in Stages 1 and 2; this function is used with default parameters, but control parameters can be supplied by the user.

After determining a critical value for a given testing problem via the function `critical.value`, it is recommended that: (i) the user applies the function `size` to compute the size of the test corresponding to the critical value obtained; and (ii) to check whether the size obtained does coincide with (or is close to) the targeted level of significance (that is `alpha`). If (ii) is not the case, this is an indication of numerical issues, which potentially can be avoided by changing the input parameters responsible for the accuracy of the computations.

Usage

```
critical.value(alpha, R, X, hcmethod, restr.cov, Mp, M1, M2,
N0 = NULL, N1 = NULL, N2 = NULL, tol = 1e-08,
control.1 = list("reltol" = 1e-02, "maxit" = dim(X)[1]*20),
control.2 = list("reltol" = 1e-03, "maxit" = dim(X)[1]*30),
cores = 1, lower = 0, eps.close = .0001, lim = 30000, acc = 0.001,
size.tol = .001, maxit = 25, as.tol = 1e-08)
```

Arguments

`alpha`	Significance level. A real number in the interval $(0, 1)$.
`R`	The restriction matrix. `critical.value` computes the (smallest) size-controlling critical value for a test of the hypothesis $R\beta = r$. `R` needs to be of full row rank, and needs to have the same number of columns as `X`.
`X`	The design matrix `X` needs to be of full column rank. The number of columns of `X` must be smaller than the number of rows of `X`.
`hcmethod`	Integer in [-1, 4]. Determines the method applied in the construction of the covariance estimator used in the test statistic. The value -1 corresponds to unadjusted (i.e., classical) F statistic without df adjustment; the value 0 corresponds to the HC0 estimator; ...; the value 4 corresponds to the HC4 estimator. Note that

in case `restr.cov` is TRUE the null-restricted versions of the covariance estimators are computed. Cf. Pötscher and Preinerstorfer (2021) and the references there for details.

`restr.cov` TRUE or FALSE. Covariance matrix estimator based on null-restricted (TRUE) or unrestricted (FALSE) residuals.

`Mp` This input is used in Algorithm 1 or 2, respectively. `Mp` is a positive integer (should be chosen large, e.g., 50000; but the feasibility depends on the dimension of `X`, etc). `Mp` determines M_0 in Algorithm 1 or 2 (i.e., `A`), respectively, that is, the number of initial values chosen in Stage 0 of that algorithm. The way initial values (i.e., the sets of variance covariance matrices Σ_j in Stage 0 of the algorithm; the diagonal entries of each Σ_j sum up to 1) are chosen is as follows:

1. If $q = 1$ and $lower = 0$, one of the initial values Σ_j is a matrix which maximizes the expectation of the quadratic form $y \mapsto y'\Sigma^{1/2}A_C\Sigma^{1/2}y$ under an n-variate standard normal distribution. Here, A_C is a matrix that is defined Pötscher and Preinerstorfer (2021). If diagonal entries of this maximizer are 0, then they are replaced by the value of `eps.close` (and the other values are adjusted so that the diagonal sums up to 1).
2. One starting value Σ_j is a diagonal matrix with constant diagonal entries.
3. If `lower` is zero, then (i) $\lceil Mp/4 \rceil - 1$ covariance matrices Σ_j are drawn by sampling their diagonals $\tau_1^2, ..., \tau_n^2$ from a uniform distribution on the unit simplex in R^n; and (ii) the remaining $M_p - (\lceil Mp/4 \rceil - 1)$ covariance matrices Σ_j are each drawn by first sampling a vector $(t_1, ..., t_n)'$ from a uniform distribution on the unit simplex in R^n, and by then obtaining the diagonal $\tau_1^2, ..., \tau_n^2$ of Σ_j via $(t_1^2, ..., t_n^2)/\sum_{i=1}^n t_i^2$. If `lower` is nonzero, then the initial values are drawn analogously, but from a uniform distribution on the subset of the unit simplex in R^n corresponding to the restriction imposed by the lower bound `lower`.
4. n starting values equal to covariance matrices with a single dominant diagonal entry and all other diagonal entries constant. The size of the dominant diagonal entry is regulated via the input parameters `eps.close` and `lower`. In case `lower` is nonzero, the size of the dominant diagonal entry equals $1 - (n-1) * (lower + eps.close)$. In case `lower` is zero, the size of the dominant diagonal entry equals $1 - eps.close$.

`M1` This input is used in Algorithm 1 or 2, respectively. A positive integer (should be chosen large, e.g., 500; but the feasibility depends on the dimension of `X`, etc). Corresponds to M_1 in the description of Algorithm 1 and 2 in Pötscher and Preinerstorfer (2021). M1 must not exceed Mp.

`M2` This input is used in Algorithm 1 or 2, respectively. A positive integer. Corresponds to M_2 in the description of Algorithm 1 and 2 in Pötscher and Preinerstorfer (2021). M2 must not exceed M1.

`N0` This input is needed in Algorithm 2. Only used in case $q > 1$ (i.e., when Algorithm 2 is used). A positive integer. Corresponds to N_0 in the description of Algorithm 2 in Pötscher and Preinerstorfer (2021).

`N1` This input is needed in Algorithm 2. Only used in case $q > 1$ (i.e., when Algorithm 2 is used). A positive integer. Corresponds to N_1 in the description of Algorithm 2 in Pötscher and Preinerstorfer (2021). N1 should be greater than N0.

`N2` This input is needed in Algorithm 2. Only used in case $q > 1$ (i.e., when Algorithm 2 is used). A positive integer. Corresponds to N_2 in the description of Algorithm 2 in Pötscher and Preinerstorfer (2021). N2 should be greater than N1.

`tol` This input is used in Algorithm 1 or 2, respectively. (Small) positive real number. Tolerance parameter used in checking invertibility of the covariance matrix in the test statistic. Default is 1e-08.

`control.1` This input is used in Algorithm 1 or 2, respectively. Control parameters passed to the `constrOptim` function in Stage 1 of Algorithm 1 or 2, respectively. Default is `control.1 = list("reltol" = 1e-02,"maxit" = dim(X)[1]*20)`.

`control.2` This input is used in Algorithm 1 or 2, respectively. Control parameters passed to the `constrOptim` function in Stage 2 of Algorithm 1 or 2, respectively. Default is `control.2 = list("reltol" = 1e-03,"maxit" = dim(X)[1]*30)`.

`cores` The number of CPU cores used in the (parallelized) computations. Default is 1. Parallelized computation is enabled only if the compiler used to build **hrt** supports OpenMP.

`lower` Number in $[0, n^{-1})$ (note that the diagonal of Σ is normalized to sum up to 1; if lower > 0, then lower corresponds to what is denoted τ_* in Pötscher and Preinerstorfer (2021)). `lower` specifies a lower bound on each diagonal entry of the (normalized) covariance matrix in the covariance model for which the user wants to obtain a critical value that achieves size control. If this lower bound is nonzero (which is the non-standard choice), then the size is only computed over all covariance matrices, which are restricted such that their minimal diagonal entry is not smaller than `lower`. The relevant optimization problems in Algorithm 1 and 2 are then carried out only over this restricted set of covariance matrices. The size will then in general depend on `lower`. See the relevant discussions concerning restricted heteroskedastic covariance models in Pötscher and Preinerstorfer (2021). Default is 0, which is the recommended choice, unless there are strong reasons implying a specific lower bound on the variance in a given application.

`eps.close` (Small) positive real number. This determines the size of the dominant entry in the choice of the initial values as discussed in the description of the input `Mp` above. Default is 1e-4.

`lim` This input is needed in Algorithm 1. Only used in case $q = 1$ (i.e., when Algorithm 1 is used). Input parameter for the function `davies`. Default is 30000.

`acc` This input is needed in Algorithm 1. Only used in case $q = 1$ (i.e., when Algorithm 1 is used). Input parameter for the function `davies`. Default is 1e-3.

`size.tol` (Small) positive real number. ϵ in Algorithm 3. Default is 1e-3.

`maxit` Maximum number of iterations in the while loop of Algorithm 3. Default is 25.

`as.tol` (Small) positive real number. Tolerance parameter used in checking rank conditions for verifying Assumptions 1, 2, and for checking a non-constancy condition on the test statistic in case `hcmethod` is not -1 and `restr.cov` is TRUE. `as.tol` is also used in the rank computations required for computing lower bounds for size-controlling critical values. Furthermore, `as.tol` is used in checking the sufficient conditions for existence of a size-controlling critical value provided in Pötscher and Preinerstorfer(2021). Default is 1e-08.

Details

For details see the relevant sections in Pötscher and Preinerstorfer (2021), in particular the description of Algorithms 1 and 2 in the Appendix.

Value

The output of `critical.value` is the following:

`critical.value` The critical value obtained by Algorithm 3.

`approximate.size`
The approximate size of the test based on the returned critical value.

`iter` The number of iterations performed. If `iter` is smaller than `maxit`, then the algorithm determined because the required level of accuracy was achieved.

References

Pötscher, B. M. and Preinerstorfer, D. (2021). Valid Heteroskedasticity Robust Testing. <arXiv:2104.12597>

See Also

`davies`, `constrOptim`.

Examples

```
#critical value for the classical (uncorrected) F-test in a location model
#with unrestricted heteroskedasticity

#it is known that (in this very special case) the conventional critical value
#C <- qt(.975, df = 9)^2
#is size-controlling (thus the resulting size should be 5% (approximately))

R <- matrix(1, nrow = 1)
X <- matrix(rep(1, length = 10), nrow = 10, ncol = 1)
hcmethod <- -1
restr.cov <- FALSE
Mp <- 1000
M1 <- 5
M2 <- 1

#here, the parameters are chosen such that the run-time is low
#to guarantee a high accuracy level in the computation,
#Mp, M1 and M2 should be chosen much higher

critical.value(alpha = .05, R, X, hcmethod, restr.cov, Mp, M1, M2)
```

`size` *Computing the Size of Heteroskedasticity Robust Tests*

Description

This function provides an implementation of Algorithm 1 (if $q = 1$) or 2 (if $q > 1$), respectively, in Pötscher and Preinerstorfer (2021). Which of the two algorithms is applied is automatically determined as a function of q.

The user is referred to the just-mentioned article for definitions, a detailed description of the problem solved the algorithms, and for a detailed description of the algorithms themselves.

Algorithm 1 is based on the function `davies` from the package **CompQuadForm**. The parameters `lim` and `acc` for `davies` can be supplemented by the user. Algorithms 1 and 2 are implemented using the function `constrOptim` from **stats** in Stages 1 and 2; this function is used with default parameters, but control parameters can be supplied by the user.

Usage

```
size(C, R, X, hcmethod, restr.cov, Mp, M1, M2,
N0 = NULL, N1 = NULL, N2 = NULL, tol = 1e-08,
control.1 = list("reltol" = 1e-02, "maxit" = dim(X)[1]*20),
control.2 = list("reltol" = 1e-03, "maxit" = dim(X)[1]*30),
cores = 1, lower = 0, eps.close = .0001, lim = 30000, acc = 0.001,
levelCl = 0, LBcheck = FALSE, as.tol = 1e-08)
```

Arguments

`C`	Critical value. A positive real number (for negative critical values the size of the test equals 1).
`R`	The restriction matrix. `size` computes the size of a test for the hypothesis $R\beta = r$. `R` needs to be of full row rank, and needs to have the same number of columns as `X`.
`X`	The design matrix `X` needs to be of full column rank. The number of columns of `X` must be smaller than the number of rows of `X`.
`hcmethod`	Integer in [-1, 4]. Determines the method applied in the construction of the covariance estimator used in the test statistic. The value -1 corresponds to the unadjusted (i.e., classical) F statistic without df adjustment; the value 0 corresponds to the HC0 estimator; ...; the value 4 corresponds to the HC4 estimator. Note that in case `restr.cov` is TRUE the null-restricted versions of the covariance estimators are computed. Cf. Pötscher and Preinerstorfer (2021) and the references there for details.
`restr.cov`	TRUE or FALSE. Covariance matrix estimator based on null-restricted (TRUE) or unrestricted (FALSE) residuals.
`Mp`	A positive integer (should be chosen large, e.g., 50000; but the feasibility depends on the dimension of `X`, etc). `Mp` determines M_0 in Algorithm 1 or 2,

respectively, that is, the number of initial values chosen in Stage 0 of that algorithm. The way initial values (i.e., the sets of variance covariance matrices Σ_j in Stage 0 of the algorithm; the diagonal entries of each Σ_j sum up to 1) are chosen is as follows:

1. If $q = 1$ and $lower = 0$, one of the initial values Σ_j is a matrix which maximizes the expectation of the quadratic form $y \mapsto y'\Sigma^{1/2}A_C\Sigma^{1/2}y$ under an n-variate standard normal distribution. Here, A_C is a matrix that is defined Pötscher and Preinerstorfer (2021). If diagonal entries of this maximizer are 0, then they are replaced by the value of `eps.close` (and the other values are adjusted so that the diagonal sums up to 1).
2. One starting value Σ_j is a diagonal matrix with constant diagonal entries.
3. If `lower` is zero, then (i) $\lceil Mp/4 \rceil - 1$ covariance matrices Σ_j are drawn by sampling their diagonals $\tau_1^2, ..., \tau_n^2$ from a uniform distribution on the unit simplex in R^n; and (ii) the remaining $M_p - (\lceil Mp/4 \rceil - 1)$ covariance matrices Σ_j are each drawn by first sampling a vector $(t_1, ..., t_n)'$ from a uniform distribution on the unit simplex in R^n, and by then obtaining the diagonal $\tau_1^2, ..., \tau_n^2$ of Σ_j via $(t_1^2, ..., t_n^2)/\sum_{i=1}^n t_i^2$. If `lower` is nonzero, then the initial values are drawn analogously, but from a uniform distribution on the subset of the unit simplex in R^n corresponding to the restriction imposed by the lower bound `lower`.
4. n starting values equal to covariance matrices with a single dominant diagonal entry and all other diagonal entries constant. The size of the dominant diagonal entry is regulated via the input parameters `eps.close` and `lower`. In case `lower` is nonzero, the size of the dominant diagonal entry equals $1 - (n-1) * (lower + eps.close)$. In case `lower` is zero, the size of the dominant diagonal entry equals $1 - eps.close$.
5. If `levelCl` is nonzero (see the description of `levelCl` below for details concerning this input), then one further initial value may be obtained by: (i) checking whether `C` exceeds 5 times the critical value C_H, say, for which the rejection probability under homoskedasticity equals $1 - levelCl$; and (ii) if this is the case, running the function `size` (with the same input parameters, but with `levelCl` set to 0 and `M2` set to 1) on the critical value C_H, and then using the output `second.stage.parameter` as a further initial value.

`M1` A positive integer (should be chosen large, e.g., 500; but the feasibility depends on the dimension of `X`, etc). Corresponds to M_1 in the description of Algorithm 1 and 2 in Pötscher and Preinerstorfer (2021). M1 must not exceed Mp.

`M2` A positive integer. Corresponds to M_2 in the description of Algorithm 1 and 2 in Pötscher and Preinerstorfer (2021). M2 must not exceed M1.

`N0` Only used in case $q > 1$ (i.e., when Algorithm 2 is used). A positive integer. Corresponds to N_0 in the description of Algorithm 2 in Pötscher and Preinerstorfer (2021).

`N1` Only used in case $q > 1$ (i.e., when Algorithm 2 is used). A positive integer. Corresponds to N_1 in the description of Algorithm 2 in Pötscher and Preinerstorfer (2021). N1 should be greater than N0.

`N2` Only used in case $q > 1$ (i.e., when Algorithm 2 is used). A positive integer. Corresponds to N_2 in the description of Algorithm 2 in Pötscher and Preinerstorfer (2021). N2 should be greater than N1.

`tol` (Small) positive real number. Tolerance parameter used in checking invertibility of the covariance matrix in the test statistic. Default is 1e-08.

`control.1` Control parameters passed to the `constrOptim` function in Stage 1 of Algorithm 1 or 2, respectively. Default is `control.1 = list("reltol" = 1e-02,"maxit" = dim(X)[1]*20)`.

`control.2` Control parameters passed to the `constrOptim` function in Stage 2 of Algorithm 1 or 2, respectively. Default is `control.2 = list("reltol" = 1e-03,"maxit" = dim(X)[1]*30)`.

`cores` The number of CPU cores used in the (parallelized) computations. Default is 1. Parallelized computation is enabled only if the compiler used to build **hrt** supports OpenMP.

`lower` Number in $[0, n^{-1})$ (note that the diagonal of Σ is normalized to sum up to 1; if lower > 0, then lower corresponds to what is denoted τ_* in Pötscher and Preinerstorfer (2021)). `lower` specifies a lower bound on each diagonal entry of the (normalized) covariance matrix in the covariance model for which the user wants to compute the size. If this lower bound is nonzero, then the size is only computed over all covariance matrices, which are restricted such that their minimal diagonal entry is not smaller than `lower`. The relevant optimization problems in Algorithm 1 and 2 are then carried out only over this restricted set of covariance matrices. The size will then in general depend on `lower`. See the relevant discussions concerning restricted heteroskedastic covariance models in Pötscher and Preinerstorfer (2021). Default is 0, which is the recommended choice, unless there are strong reasons implying a specific lower bound on the variance in a given application.

`eps.close` (Small) positive real number. This determines the size of the dominant entry in the choice of the initial values as discussed in the description of the input `Mp` above. Default is 1e-4.

`lim` This input is needed in Algorithm 1. Only used in case $q = 1$ (i.e., when Algorithm 1 is used). Input parameter for the function `davies`. Default is 30000.

`acc` This input is needed in Algorithm 1. Only used in case $q = 1$ (i.e., when Algorithm 1 is used). Input parameter for the function `davies`. Default is 1e-3.

`levelCl` Number in $[0, 1)$. This enters via the choice of the initial values as discussed in the input `Mp` above. `levelCl` should be used in case `C` is unusually large. In this case, the additional set of starting values provided may help to increase the accuracy of the size computation. Default is 0.

`LBcheck` Either FALSE (default) or TRUE. If TRUE, then `C` is compared to the theoretical lower bounds on size-controlling critical values in Pötscher and Preinerstorfer (2021). If the supplemented `C` is smaller than the respective lower bound, theoretical results imply that the size equals 1 and the function `size` is halted.

`as.tol` (Small) positive real number. Tolerance parameter used in checking rank conditions for verifying Assumptions 1, 2, and for checking a non-constancy condition on the test statistic in case `hcmethod` is not -1 and `restr.cov` is TRUE. Furthermore, `as.tol` is used in the rank computations required for computing lower bounds for size-controlling critical values (in case `LBcheck` is TRUE or `levelCl` is nonzero). Default is 1e-08.

Details

For details see the relevant sections in Pötscher and Preinerstorfer (2021), in particular the description of Algorithms 1 and 2 in the Appendix.

Value

The output of `size` is the following:

`starting.parameters`
: The rows of this matrix are the initial values (diagonals of covariance matrices) that were used in Stage 1 of the algorithm, and which were chosen from the pool of initial values in Stage 0.

`starting.rejection.probs`
: The null-rejection probabilities corresponding to the initial values used in Stage 1.

`first.stage.parameters`
: The rows of this matrix are the parameters (diagonals of covariance matrices) that were obtained in Stage 1 of the algorithm.

`first.stage.rejection.probs`
: The null-rejection probabilities corresponding to the `first.stage.parameters`.

`second.stage.parameters`
: The rows of this matrix are the parameters (diagonals of covariance matrices) that were obtained in Stage 2 of the algorithm.

`second.stage.rejection.probs`
: The null-rejection probabilities corresponding to the `second.stage.parameters`.

`convergence`
: Convergence codes returned from `constrOptim` in Stage 2 of the algorithm for each initial value.

`size`
: The size computed by the algorithm, i.e., the maximum of the `second.stage.rejection.probs`.

References

Pötscher, B. M. and Preinerstorfer, D. (2021). Valid Heteroskedasticity Robust Testing. <arXiv:2104.12597>

See Also

`davies`, `constrOptim`.

Examples

```
#size of the classical (uncorrected) F-test in a location model
#with conventional t-critical value (5% level)

#it is known that (in this very special case) the conventional critical value
#is size-controlling (i.e., the resulting size should be 5% (approximately))

C <- qt(.975, df = 9)^2
R <- matrix(1, nrow = 1)
X <- matrix(rep(1, length = 10), nrow = 10, ncol = 1)
```

```
hcmethod <- -1
restr.cov <- FALSE
Mp <- 100
M1 <- 5
M2 <- 1

#here, the parameters are chosen such that the run-time is low
#to guarantee a high accuracy level in the computation,
#Mp, M1 and M2 should be chosen much higher

size(C, R, X, hcmethod, restr.cov, Mp, M1, M2)
```

`test.stat` *Computation of the test statistics*

Description

This function computes the test statistics T_{uc}, T_{Het} (with HC0-HC4 weights), $\tilde{T}_{uc}$, or $\tilde{T}_{Het}$ (with HC0R-HC4R weights) as defined in Pötscher and Preinerstorfer (2021).

Usage

```
test.stat(y, R, r, X, hcmethod, restr.cov, tol = 1e-08, cores = 1)
```

Arguments

`y`	Either an observation vector, or a matrix the columns of which are observation vectors. The number of rows of an observation vector must coincide with the number of rows of the design matrix `X`.
`R`	The restriction matrix. `test.stat` computes a test statistic for the hypothesis $R\beta = r$. `R` needs to be of full row rank, and needs to have the same number of columns as `X`.
`r`	The restriction vector. `test.stat` computes a test statistic for the hypothesis $R\beta = r$. `r` needs to be a vector with the same number of coordinates as the number of rows of `R`.
`X`	The design matrix `X` needs to be of full column rank. The number of columns of `X` must be smaller than the number of rows of `X`.
`hcmethod`	Integer in [-1, 4]. Determines the method applied in the construction of the covariance estimator used in the test statistic. The value -1 corresponds to unadjusted (i.e., classical) F statistic without df adjustment; the value 0 corresponds to the HC0 estimator; ...; the value 4 corresponds to the HC4 estimator. Note that in case `restr.cov` is TRUE the null-restricted versions of the covariance estimators are computed. Cf. Pötscher and Preinerstorfer (2021) and the references there for details.
`restr.cov`	TRUE or FALSE. Covariance matrix estimator based on null-restricted (TRUE) or unrestricted (FALSE) residuals.

`tol`	(Small) positive real number. Tolerance parameter used in checking invertibility of the covariance matrix in the test statistic. Default is 1e-08.
`cores`	The number of CPU cores used in the (parallelized) computations. Default is 1. Parallelized computation is enabled only if the compiler used to build **hrt** supports OpenMP.

Details

For details concerning the test statistics please see the relevant sections in Pötscher and Preinerstorfer (2021) .

Value

The function returns a list consisting of:

`test.val`	Either a vector the entries of which correspond to the values of the test statistic evaluated at each column of the input matrix `y`, or, if `y` is a vector, the test statistic evaluated at `y`.

References

Pötscher, B. M. and Preinerstorfer, D. (2021). Valid Heteroskedasticity Robust Testing. <arXiv:2104.12597>

Examples

```
n <- 20
y <- rnorm(n)
X <- cbind(rep(1, length = n), rnorm(n))
R <- matrix(1, nrow = 1, ncol = 2)
r <- 0
hcmethod <- 4
restr.cov <- FALSE
test.stat(y, R, r, X, hcmethod, restr.cov)
```

Index

constrOptim, *3*, *5–7*, *9*, *10*
critical.value, 3

davies, *3*, *6*, *7*, *10*

hrt *(*hrt-package*)*, 2
hrt-package, 2

size, 7

test.stat, 11