Package 'robust2sls'

August 15, 2022

Type Package

Title Outlier Robust Two-Stage Least Squares Inference and Testing

Version 0.2.1

Description An implementation of easy tools for outlier robust inference in two-stage least squares (2SLS) models. The user specifies a reference distribution against which observations are classified as outliers or not. After removing the outliers, adjusted standard errors are automatically provided. Furthermore, several statistical tests for the false outlier detection rate can be calculated. The outlier removing algorithm can be iterated a fixed number of times or until the procedure converges. The algorithms and robust inference are described in more detail in Jiao (2019) <https://drive.google.com/file/d/1qPxDJnLlzLqdk94X9wwVASptf1MPpI2w/view>.

URL https://github.com/jkurle/robust2sls

BugReports https://github.com/jkurle/robust2sls/issues

License GPL-3

Encoding UTF-8

Suggests covr,
datasets,
doFuture,
doParallel,
future,
ggplot2,
grDevices,
ivgets,
knitr,
parallel,
rmarkdown,
testthat,
utils

RoxygenNote 7.1.2

Config/testthat/edition 3

Imports doRNG,
exactci,
foreach,
ivreg,
MASS,

mathjaxr,
pracma,
stats

VignetteBuilder knitr

Depends R (>= 2.10)

RdMacros mathjaxr

R topics documented:

`robust2sls-package` *robust2sls: A package for outlier robust 2SLS inference and testing*

Description

The robust2sls package provides two main functionalities. First, it implements an algorithm for determining whether an observations is an outlier based on its standardized residual and re-estimation based on the sub-sample excluding all outliers. This procedure is often used in empirical research to show that the results are not driven by outliers. This package has implemented the algorithm in various forms and the user can select between different initial estimators and how often the algorithm is iterated. The statistical inference is adapted to account for potential false positives (classifying observations as outliers even though they are not).

Second, the robust2sls package provides easy-to-use statistical tests on whether the difference between the original and the outlier-robust estimates is statistically significant. Furthermore, several different statistical tests are implemented to test whether the sample actually contains outliers.

`beta_hausman` *Calculates a Hausman test on the difference between robust and full sample estimates*

Description

Calculates a Hausman test on the difference between robust and full sample estimates

Usage

```
beta_hausman(robust2sls_object, iteration, subset = NULL, fp = FALSE)
```

Arguments

`robust2sls_object`	An object of class `"robust2sls"`.
`iteration`	An integer > 0 specifying the iteration step for which parameters to calculate corrected standard errors.
`subset`	A vector of numeric indices or strings indicating which coefficients to include in the Hausman test. `NULL` uses the whole vector of coefficients.
`fp`	A logical value whether the fixed point asymptotic variance (TRUE) or the exact iteration asymptotic variance should be used (FALSE).

Details

Argument `fp` determines whether the fixed point asymptotic variance should be used. This argument is only respected if the specified `iteration` is one of the iterations after the algorithm converged.

Value

`beta_hausman` returns a matrix with the value of the Hausman test statistic and its corresponding p-value. The attribute `"type of avar"` records which asymptotic variance has been used (the specific iteration or the fixed point). The attribute `"coefficients"` stores the names of the coefficients that were included in the Hausman test.

`beta_inf` *Calculates valid se for coefficients under H0 of no outliers*

Description

Calculates valid se for coefficients under H0 of no outliers

Usage

```
beta_inf(robust2sls_object, iteration = 1, exact = FALSE, fp = FALSE)
```

Arguments

`robust2sls_object`	An object of class `"robust2sls"`.
`iteration`	An integer > 0 specifying the iteration step for which parameters to calculate corrected standard errors.
`exact`	A logical value indicating whether the actually detected share of outliers (TRUE) or the theoretical share (FALSE) should be used.
`fp`	A logical value whether the fixed point standard error correction (TRUE) or the exact iteration correction should be computed (FALSE).

Details

Argument `iteration` specifies which iteration of the robust structural parameter estimates should be calculated. Iteration `1` refers to the first robust estimate. Iteration `0` is not a valid argument since it is the baseline estimate, which is not robust.

The parameter `exact` does not matter much under the null hypothesis of no outliers since the detected share will converge to the theoretical share. Under the alternative, this function should not be used.

Argument `fp` determines whether the fixed point standard error correction should be computed. This argument is only respected if the specified `iteration` is one of the iterations after the algorithm converged.

Value

`beta_inf` returns the corrected standard errors for the structural parameters. These are valid under the null hypothesis of no outliers in the sample. For comparison, the uncorrected standard errors are also reported.

`beta_inf_correction` *Calculates the correction factor for inference under H0 of no outliers*

Description

Calculates the correction factor for inference under H0 of no outliers

Usage

```
beta_inf_correction(
  robust2sls_object,
  iteration = 1,
  exact = FALSE,
  fp = FALSE
)
```

Arguments

`robust2sls_object`	An object of class `"robust2sls"`.
`iteration`	An integer > 0 specifying the iteration step for which parameters to calculate corrected standard errors.
`exact`	A logical value indicating whether the actually detected share of outliers (TRUE) or the theoretical share (FALSE) should be used.
`fp`	A logical value whether the fixed point standard error correction (TRUE) or the exact iteration correction should be computed (FALSE).

Details

Argument `iteration` specifies which iteration of the robust structural parameter estimates should be calculated. Iteration `1` refers to the first robust estimate. Iteration `0` is not a valid argument since it is the baseline estimate, which is not robust.

The parameter `exact` does not matter much under the null hypothesis of no outliers since the detected share will converge to the theoretical share. Under the alternative, this function should not be used.

Argument `fp` determines whether the fixed point standard error correction should be computed. This argument is only respected if the specified `iteration` is one of the iterations after the algorithm converged.

Value

`beta_inf_correction` returns the numeric correction factor.

beta_t — *Conducts a t-test on the difference between robust and full sample estimates*

Description

Conducts a t-test on the difference between robust and full sample estimates

Usage

```
beta_t(robust2sls_object, iteration, element, fp = FALSE)
```

Arguments

robust2sls_object	An object of class "robust2sls".
iteration	An integer > 0 specifying the iteration step for which parameters to calculate corrected standard errors.
element	An index or a string to select the coefficient which is to be tested. The index should refer to the index of coefficients in the "ivreg" model object, i.e. $coefficients.
fp	A logical value whether the fixed point asymptotic variance (TRUE) or the exact iteration asymptotic variance should be used (FALSE).

Details

Argument fp determines whether the fixed point asymptotic variance should be used. This argument is only respected if the specified iteration is one of the iterations after the algorithm converged.

Value

beta_t returns a matrix with the robust and full sample estimates of beta, the t statistic on their difference, the standard error of the difference, and three p-values (two-sided, both one-sided alternatives).

beta_test_avar — *Calculates the asymptotic variance of the difference between robust and full sample estimators of the structural parameters*

Description

Calculates the asymptotic variance of the difference between robust and full sample estimators of the structural parameters

Usage

```
beta_test_avar(robust2sls_object, iteration, fp = FALSE)
```

Arguments

robust2sls_object	An object of class "robust2sls".
iteration	An integer > 0 specifying the iteration step for which parameters to calculate corrected standard errors.
fp	A logical value whether the fixed point asymptotic variance (TRUE) or the exact iteration asymptotic variance should be computed (FALSE).

Details

Argument fp determines whether the fixed point asymptotic variance should be computed. This argument is only respected if the specified iteration is one of the iterations after the algorithm converged.

Value

beta_test_avar returns a dx by dx variance-covariance matrix of the difference between the robust and full sample structural parameter estimates of the 2SLS model.

---

case_resampling *Uses nonparametric case resampling for standard errors of parameters and gauge*

Description

Uses nonparametric case resampling for standard errors of parameters and gauge

Usage

```
case_resampling(robust2sls_object, R, coef = NULL, m = NULL, parallel = FALSE)
```

Arguments

robust2sls_object	An object of class "robust2sls".
R	An integer specifying the number of resamples.
coef	A numeric or character vector specifying which structural coefficient estimates should be recorded across bootstrap replications. NULL means all coefficients are recorded.
m	A single numeric or vector of integers specifying for which iterations the bootstrap statistics should be calculated. NULL means they are calculated for all iterations that were also done in the original robust2sls_object. Character "convergence" means all bootstrap samples are run until they converge and the statistics of the first convergent iteration is recorded.
parallel	A logical value indicating whether to run the bootstrap sampling in parallel or sequentially. See Details.

Details

Argument `parallel` allows for parallel computing using the foreach package, so the user has to register a parallel backend before invoking this command.

Argument `coef` is useful if the model includes many controls whose parameters are not of interest. This can reduce the memory space needed to store the bootstrap results.

Value

`case_resampling` returns an object of class `"r2sls_boot"`. This is a list with three named elements. `$boot` stores the bootstrap results as a data frame. The columns record the different test statistics, the iteration `m`, and the number of the resample, `r`. The values corresponding to the original data is stored as `r = 0`. `$resamples` is a list of length `R` that stores the indices for each specific resample. `$original` stores the original `robust2sls_object` based on which the bootstrapping was done.

---

counttest *Count test*

Description

`counttest()` conducts a test whether the number of detected outliers deviates significantly from the expected number of outliers under the null hypothesis that there are no outliers in the sample.

Usage

```
counttest(
  robust2sls_object,
  alpha,
  iteration,
  one_sided = FALSE,
  tsmethod = c("central", "minlike", "blaker")
)
```

Arguments

`robust2sls_object`	An object of class `"robust2sls"` or a list of such objects.
`alpha`	A numeric value between 0 and 1 representing the significance level of the test.
`iteration`	An integer >= 0 or the character "convergence" that determines which iteration is used for the test.
`one_sided`	A logical value whether a two-sided test (`FALSE`) should be conducted or a one-sided test (`TRUE`) that rejects only when the number of detected outliers is above the expected number.
`tsmethod`	A character specifying the method for calculating two-sided p-values. Ignored for one-sided test.

Details

See `outlier_detection()` and `multi_cutoff()` for creating an object of class `"robust2sls"` or a list thereof.

See `exactci::poisson.exact()` for the different methods of calculating two-sided p-values.

Value

counttest() returns a data frame with the iteration (m) to be tested, the actual iteration that was tested (generally coincides with the iteration that was specified to be tested but is the convergent iteration if the fixed point is tested), the setting of the probability of exceeding the cut-off (gamma), the number of detected outliers, the expected number of outliers under the null hypothesis that there are no outliers, the type of test (one- or two-sided), the p-value, the significance level alpha, the decision, and which method was used to calculate (two-sided) p-values. The number of rows of the data frame corresponds to the length of the argument robust2sls_object.

---

count_indices *Counts the number of times each index was sampled*

Description

count_indices takes a list of indices for resampling and counts how often each index was sampled in each resample. The result is returned in two versions of a matrix where each row corresponds to a different resample and each column to one index.

Usage

```
count_indices(resamples, indices)
```

Arguments

resamples	A list of resamples, as created by nonparametric.
indices	The vector of original indices from which the resamples were drawn.

Value

count_indices returns a list with two names elements. Each element is a matrix that stores how often each observation/index was resampled (column) for each resample (row). $count_clean only has columns for observations that were available in the indices. $count_all counts the occurrence of all indices in the range of indices that were provided, even if the index was actually not available in the given indices. These are of course zero since they were not available for resampling. If the given indices do not skip any numbers, the two coincide.

---

estimate_param *Estimation of moments of the data*

Description

NOTE (12 Apr 2022): probably superseded by estimate_param_null() function taken out of testing

Usage

```
estimate_param(robust2SLS_object, iteration)
```

Arguments

`robust2SLS_object`
: An object of class `"robust2sls"` for which the moments will be calculated.

`iteration`
: An integer >= 0 specifying based on which model iteration the moments should be estimated. The model iteration affects which observations are determined to be outliers and these observations will hence be excluded during the estimation of the moments.

Details

DO NOT USE YET! `estimate_param` can be used to estimate certain moments of the data that are required for calculating the asymptotic variance of the gauge. Such moments are the covariance between the standardised first stage errors and the structural error Ω, the covariance matrix of the first stage errors Σ, the first stage parameter matrix Π, and more.

Value

`estimate_param` returns a list with a similar structure as the output of the Monte Carlo functionality generate_param. Hence, the resulting list can be given to the function gauge_avar as argument `parameters` to return an estimate of the asymptotic variance of the gauge.

Warning

The function is not yet fully developed. The estimators of the moments are at the moment not guaranteed to be consistent for the population moments. DO NOT USE!

`estimate_param_null` *Estimation of moments of the data*

Description

`estimate_param_null` can be used to estimate certain moments of the data that are required for calculating the asymptotic variance of the gauge. Such moments are the covariance between the standardised first stage errors and the structural error Ω, the covariance matrix of the first stage errors Σ, the first stage parameter matrix Π, and more.

Usage

```
estimate_param_null(robust2SLS_object)
```

Arguments

`robust2SLS_object`
: An object of class `"robust2sls"` for which the moments will be calculated.

Value

`estimate_param_null` returns a list with a similar structure as the output of the Monte Carlo functionality generate_param. Hence, the resulting list can be given to the function gauge_avar as argument `parameters` to return an estimate of the asymptotic variance of the gauge.

Warning

The function uses the full sample to estimate the moments. Therefore, they are only consistent under the null hypothesis of no outliers and estimators are likely to be inconsistent under the alternative.

`evaluate_boot` *Evaluate bootstrap results*

Description

Evaluate bootstrap results

Usage

```
evaluate_boot(r2sls_boot, iterations)
```

Arguments

`r2sls_boot`	An object of class `"r2sls_boot"`, as returned by case_resampling.
`iterations`	An integer or numeric vector with values >= 0 specifying which bootstrap results to evaluate.

Value

`evaluate_boot` returns a data frame with the bootstrap and the theoretical standard errors. Each row corresponds to a different iteration step while each column refers to the parameters whose standard errors are produced.

`extract_boot` *Extracts bootstrap results for a specific iteration*

Description

Extracts bootstrap results for a specific iteration

Usage

```
extract_boot(r2sls_boot, iteration)
```

Arguments

`r2sls_boot`	An object of class `"r2sls_boot"`, as returned by case_resampling.
`iteration`	An integer >= 0 specifying which bootstrap results to extract.

Value

`extract_boot` returns a matrix with the bootstrap results for a specific iteration.#'

gauge_avar *Asymptotic variance of gauge*

Description

gauge_avar calculates the asymptotic variance of the gauge for a given iteration using a given set of parameters (true or estimated).

Usage

```
gauge_avar(
  ref_dist = c("normal"),
  sign_level,
  initial_est = c("robustified", "saturated", "iis"),
  iteration,
  parameters,
  split
)
```

Arguments

ref_dist	A character vector that specifies the reference distribution against which observations are classified as outliers. "normal" refers to the normal distribution.
sign_level	A numeric value between 0 and 1 that determines the cutoff in the reference distribution against which observations are judged as outliers or not.
initial_est	A character vector that specifies the initial estimator for the outlier detection algorithm. "robustified" means that the full sample 2SLS is used as initial estimator. "saturated" splits the sample into two parts and estimates a 2SLS on each subsample. The coefficients of one subsample are used to calculate residuals and determine outliers in the other subsample. "iis" applies impulse indicator saturation (IIS) as implemented in ivisat.
iteration	An integer >= 0 or character "convergence" representing the iteration for which the outliers are calculated. Uses the fixed point value if set to "convergence".
parameters	A list created by generate_param or estimate_param_null that stores the parameters (true or estimated). NULL permitted if ref_dist == "normal".
split	A numeric value strictly between 0 and 1 that determines in which proportions the sample will be split. Can be NULL if initial_est == "robustified".

Details

Initial estimator "iis" uses the asymptotic variances of "robustified" 2SLS because there is no formal theory for the multi-block search.

Value

gauge_avar returns a numeric value.

gauge_covar *Asymptotic covariance of gauge*

Description

gauge_covar calculates the asymptotic covariance between two FODRs with different cut-off values s and t for a given iteration using a given set of parameters (true or estimated).

Usage

```
gauge_covar(
  ref_dist = c("normal"),
  sign_level1,
  sign_level2,
  initial_est = c("robustified", "saturated", "iis"),
  iteration,
  parameters,
  split
)
```

Arguments

ref_dist	A character vector that specifies the reference distribution against which observations are classified as outliers. "normal" refers to the normal distribution.
sign_level1	A numeric value between 0 and 1 that determines the first cutoff in the reference distribution against which observations are judged as outliers or not.
sign_level2	A numeric value between 0 and 1 that determines the second cutoff in the reference distribution against which observations are judged as outliers or not.
initial_est	A character vector that specifies the initial estimator for the outlier detection algorithm. "robustified" means that the full sample 2SLS is used as initial estimator. "saturated" splits the sample into two parts and estimates a 2SLS on each subsample. The coefficients of one subsample are used to calculate residuals and determine outliers in the other subsample. "iis" applies impulse indicator saturation (IIS) as implemented in ivisat.
iteration	An integer >= 0 or character "convergence" representing the iteration for which the outliers are calculated. Uses the fixed point value if set to "convergence".
parameters	A list created by generate_param or estimate_param_null that stores the parameters (true or estimated). NULL permitted if ref_dist == "normal".
split	A numeric value strictly between 0 and 1 that determines in which proportions the sample will be split. Can be NULL if initial_est == "robustified".

Details

Initial estimator "iis" uses the asymptotic variances of "robustified" 2SLS because there is no formal theory for the multi-block search.

Value

gauge_covar returns a numeric value.

generate_data *Random data of 2SLS model (Monte Carlo)*

Description

generate_data draws random data for a 2SLS model given the parameters.

Usage

```
generate_data(parameters, n)
```

Arguments

parameters	A list with 2SLS model parameters as created by generate_param.
n	Sample size to be drawn.

Value

generate_data returns a data frame with n rows (observations) and the following variables of the 2SLS model: dependent variable y, exogenous regressors x1, endogenous regressors x2, structural error u, outside instruments z2, first stage projection errors r1 (identical to zero) and r2.

generate_param *Parameters of 2SLS model (Monte Carlo)*

Description

By default, generate_param creates random parameters of a 2SLS model that satisfy conditions for 2SLS models, such as positive definite variance-covariance matrices. The user can also specify certain parameters directly, which are then checked for their validity.

Usage

```
generate_param(
  dx1,
  dx2,
  dz2,
  intercept = TRUE,
  beta = NULL,
  sigma = 1,
  mean_z = NULL,
  cov_z = NULL,
  Sigma2_half = NULL,
  Omega2 = NULL,
  Pi = NULL,
  seed = 42
)
```

Arguments

dx1	An integer value specifying the number of exogenous regressors. This should include the intercept if it is present in the model (see argument intercept).
dx2	An integer value specifying the number of endogenous regressors.
dz2	An integer value specifying the number of outside / excluded instruments.
intercept	A logical value (TRUE / FALSE) indicating whether the model should contain an intercept.
beta	A numeric vector of length dx1 + dx2 specifying the parameters of the structural equation.
sigma	A strictly positive numeric value specifying the standard deviation of the error in the structural model.
mean_z	A numeric vector of length dx1 + dz2 specifying the mean of the exogenous variables, x1 and z2.
cov_z	A numeric positive definite matrix specifying the variance-covariance matrix of the exogenous variables, x1 and z2.
Sigma2_half	A numeric positive definite matrix of dimension dx2 by dx2 such that its square is the variance-covariance matrix of the random first stage errors (Sigma2).
Omega2	A numeric vector of length dx1 specifying the correlation between the scaled random first stage error and the structural error.
Pi	A numeric matrix of dimension (dx1 + dz2) by (dx1 + dx2) specifying the first stage parameter matrix.
seed	An integer for setting the seed for the random number generator.

Value

generate_param returns a list with the (randomly created or user-specified) parameters that are required for drawing random data that. The parameters are generated to fulfill the 2SLS model assumptions.

$structural A list with two components storing the mean ($mean) and variance-covariance matrix ($cov) for the structural error (u), the random first stage errors (r2), and all instruments (excluding the intercept since it is not random) (z).

$params A list storing the parameters of the 2SLS model. $beta is the coefficient vector (including intercept if present) of the structural equation, $Pi the coefficient matrix of the first stage projections, $Omega2 the covariance between the structural error and the endogenous first stage errors, $Sigma2_half the square root of the variance-covariance matrix of the endogenous first stage errors, $mean_z the mean of all instruments (excluding the intercept since it is not random), $cov_z the variance-covariance matrix of the endogenous first-stage errors, $Ezz the expected value of the squared instruments.

$settings A list storing the function call ($call), whether an intercept is included in the model ($intercept), a regression formula for the model setup ($formula), and the dimensions of the regressors and instruments ($dx1, $dx2, $dz2.

$names A list storing generic names for the regressors, instruments, and errors as character vectors ($x1, $x2, $x, $z2, $z, $r, and $u).

globaltest *Global test correcting for multiple hypothesis testing*

Description

`globaltest()` uses several proportion or count tests with different cut-offs to test a global hypothesis of no outliers using the Simes (1986) procedure to account for multiple testing.

Usage

```
globaltest(tests, global_alpha)
```

Arguments

`tests`	A data frame that contains a column named `$pval` containing the different p-values for different hypothesis tests, each stored in a row.
`global_alpha`	A numeric value representing the global significance level.

Details

See Simes (1986).

Value

A list with three entries. The first entry named `$reject` contains the global rejection decision. The second entry named `$global_alpha` stores the global significance level. The third entry named `$tests` returns the input data frame `tests`, appended with two columns containing the adjusted significance level and respective rejection decision.

See Also

[proptest()], [counttest()]

iis_init *Impulse Indicator Saturation (IIS initial estimator)*

Description

Impulse Indicator Saturation (IIS initial estimator)

Usage

```
iis_init(
  data,
  formula,
  gamma,
  t.pval = gamma,
  do.pet = FALSE,
  normality.JarqueB = NULL,
  turbo = FALSE,
```

```
  overid = NULL,
  weak = NULL
)
```

Arguments

data	A dataframe.
formula	A formula in the format y ~ x1 + x2 | x1 + z2 where y is the dependent variable, x1 are the exogenous regressors, x2 the endogenous regressors, and z2 the outside instruments.
gamma	A numeric value between 0 and 1 representing the significance level used for two-sided significance t-test on the impulse indicators. Corresponds to the probability of falsely classifying an observation as an outlier.
t.pval	A numeric value between 0 and 1 representing the significance level for the Parsimonious Encompassing Test (PET).
do.pet	logical. If TRUE, then a Parsimonious Encompassing Test (PET) against the GUM is undertaken at each regressor removal for the joint significance of all the deleted regressors along the current path. If FALSE (default), then a PET is not undertaken at each regressor removal. By default, the numeric value is the same as that of t.pval
normality.JarqueB	NULL (the default) or a value between 0 and 1. In the latter case, a test for non-normality is conducted using a significance level equal to normality.JarqueB. If NULL, then no test for non-normality is conducted
turbo	logical. If TRUE, then (parts of) paths are not searched twice (or more) unnecessarily, thus yielding a significant potential for speed-gain. However, the checking of whether the search has arrived at a point it has already been comes with a slight computational overhead. Accordingly, if turbo=TRUE, then the total search time might in fact be higher than if turbo=FALSE. This happens if estimation is very fast, say, less than quarter of a second. Hence the default is FALSE
overid	NULL if no Sargan test of overidentifying restrictions should be used as a diagnostic check for model selection or a numeric value between 0 and 1. In the latter case, the test is conducted using this value as the significance level.
weak	NULL if no weak instrument F-test on the first stage should be used as a diagnostic check for model selection or a numeric value between 0 and 1. In the latter case, the test is conducted using this value as the significance level.

Value

iis_init returns a list with five elements. The first four are vectors whose length equals the number of observations in the data set. Unlike the residuals stored in a model object (usually accessible via model$residuals), it does not ignore observations where any of y, x or z are missing. It instead sets their values to NA.

The first element is a double vector containing the residuals for each observation based on the model estimates. The second element contains the standardised residuals, the third one a logical vector with TRUE if the observation is judged as not outlying, FALSE if it is an outlier, and NA if any of y, x, or z are missing. The fourth element of the list is an integer vector with three values: 0 if the observations is judged to be an outlier, 1 if not, and -1 if missing. The fifth and last element stores the ivreg model object based on which the four vectors were calculated.

Note

IIS runs multiple models, similar to `saturated_init` but with multiple block search. These intermediate models are not recorded. For simplicity, the element `$model` of the returned list stores the full sample model result, identical to `robustified_init`.

`mc_grid` *Monte Carlo simulations parameter grid*

Description

WARNING: not for average user - function not completed yet

Usage

```
mc_grid(
  M,
  n,
  seed,
  parameters,
  formula,
  ref_dist,
  sign_level,
  initial_est,
  iterations,
  convergence_criterion = NULL,
  max_iter = NULL,
  shuffle = FALSE,
  shuffle_seed = 10,
  split = 0.5,
  path = FALSE,
  verbose = FALSE
)
```

Arguments

`M`	Number of replications.
`n`	Sample size for each replication.
`seed`	Random seed for the iterations.
`parameters`	A list as created by generate_param that specifies the true model.
`formula`	A formula that specifies the 2SLS model to be estimated. The format has to follow `y ~ x1 + x2 | x1 + z2`, where `y` is the dependent variable, `x1` are the exogenous regressors, `x2` the endogenous regressors, and `z2` the outside instruments.
`ref_dist`	A character vector that specifies the reference distribution against which observations are classified as outliers. `"normal"` refers to the normal distribution.
`sign_level`	A numeric value between 0 and 1 that determines the cutoff in the reference distribution against which observations are judged as outliers or not.

initial_est
: A character vector that specifies the initial estimator for the outlier detection algorithm. "robustified" means that the full sample 2SLS is used as initial estimator. "saturated" splits the sample into two parts and estimates a 2SLS on each subsample. The coefficients of one subsample are used to calculate residuals and determine outliers in the other subsample. "user" allows the user to specify a model based on which observations are classified as outliers.

iterations
: An integer >= 0 that specifies how often the outlier detection algorithm is iterated and for which summary statistics will be calculated. The value 0 means that outlier classification based on the initial estimator is done. Alternatively, the character "convergence" for iteration until convergence.

convergence_criterion
: A numeric value that determines whether the algorithm has converged as measured by the L2 norm of the difference in coefficients between the current and the previous iteration. Only used when argument iterations is set to "convergence".

max_iter
: A numeric value >= 1 or NULL. If iterations = "convergence" is chosen, then the algorithm is stopped after at most max_iter iterations. If also a convergence_criterion is chosen then the algorithm stops when either the criterion is fulfilled or the maximum number of iterations is reached.

shuffle
: A logical value or NULL. initial_est == "saturated". If TRUE then the sample is shuffled before creating the subsamples.

shuffle_seed
: An integer value that will set the seed for shuffling the sample or NULL. Only used if initial_est == "saturated" and shuffle == TRUE.

split
: A numeric value strictly between 0 and 1 that determines in which proportions the sample will be split.

path
: A character string or FALSE. The simulation grid can save the individual results of each different entry in the grid to this location. Individual results not saved if argument set to FALSE.

verbose
: A logical value whether any messages should be printed.

Details

mc_grid runs Monte Carlo simulations to assess the performance of the theory of the gauge, simple proportion tests, and count tests.

Value

mc_grid returns a data frame with the results of the Monte Carlo experiments. Each row corresponds to a specific simulation setup. The columns record the simulation setup and its results. Currently, the average proportion of detected outliers ("mean_gauge") and their variance ("var_gauge") are being recorded. Moreover, the theoretical asymptotic variance ("avar") and the ratio of simulated to theoretical variance - adjusted by the sample size - are calculated ("var_ratio"). Furthermore, tentative results of size and power for the tests are calculated.

Details

The following arguments can also be supplied as a vector of their type: n, sign_level, initial_est, and split. This makes the function estimate all possible combinations of the arguments. Note that the initial estimator "robustified" is not affected by the argument split and hence is not varied in this case.

For example, specifying n = c(100,1000) and sign_level = c(0.01,0.05) estimates four Monte Carlo experiments with the four possible combinations of the parameters.

The path argument allows users to store the M replication results for all of the individual Monte Carlo simulations that are part of the grid. The results are saved both as .Rds and .csv files. The file name is indicative of the simulation setting.

---

multi_cutoff *Multiple models, varying cut-off*

Description

multi_cutoff() runs several outlier detection algorithms that differ in the value of the cut-off that determines whether an observation is classified as an outlier or not.

Usage

```
multi_cutoff(gamma, ...)
```

Arguments

gamma	A numeric vector representing the probability of falsely classifying an observation as an outlier. One setting of the algorithm per element of gamma is being run.
...	Arguments for specifying the other settings of the outlier detection algorithm, outlier_detection.

Details

mutli_cutoff uses the foreach and future packages to run several models at the same time in parallel. This means the user has to register a backend and thereby determine how the code should be executed. The default is sequential, i.e. not in parallel. See future::plan() for details.

Value

A list containing the robust2sls objects, one per setting of gamma. The length of the list therefore corresponds to the length of the vector gamma.

---

mvn_sup *Multivariate normal supremum simulation*

Description

mvn_sup simulates the distribution of the supremum of the specified multivariate normal distribution by drawing repeatedly from the multivariate normal distribution and calculating the maximum of each vector.

Usage

```
mvn_sup(n, mu, Sigma, seed = NULL)
```

Arguments

n	An integer determining the number of draws from the multivariate normal distribution.
mu	A numeric vector representing the mean of the multivariate normal distribution.
Sigma	A numeric matrix representing the variance-covariance matrix of the mutlivariate normal distribution.
seed	An integer setting the random seed or NULL if it should not be set.

Value

mvn_sup returns a vector of suprema of length n.

---

nonparametric *Create indices for nonparametric bootstrap*

Description

nonparametric is used for nonparametric resampling, for example nonparametric case or error/residual resampling. The function takes a vector of indices that correspond to the indices of observations that should be used in the resampling procedure.

Usage

```
nonparametric(
  indices,
  R,
  size = length(indices),
  replacement = TRUE,
  seed = NULL
)
```

Arguments

indices	A vector of indices (integer) from which to sample.
R	An integer specifying the number of resamples.
size	An integer specifying the size of the resample. Standard bootstrap suggests to resample as many datapoints as in the original sample, which is set as the default.
replacement	A logical value whether to sample with (TRUE) or without (FALSE) replacement. Standard bootstrap suggests to resample with replacement, which is set as the default.
seed	NULL if seed should not be set explicitly or an integer to which the seed is set. Since this function is usually used inside other functions, it might not be desirable to set a seed explicitly.

Value

nonparametric returns a list of length R containing vectors with the resampled indices.

nonparametric_resampling *Nonparametric resampling from a data frame*

Description

Nonparametric resampling from a data frame

Usage

```
nonparametric_resampling(df, resample)
```

Arguments

df	Data frame containing observations to be sampled from.
resample	A vector of indices that extract the observations from the data frame.

Details

The input to the `resample` argument could for example be generated as one of the elements in the list generated by the command nonparametric.

The input to the `df` argument would be the original data frame for case resampling. For error/residual resampling, it would be a data frame containing the residuals from the model.

Value

`nonparametric_resampling` returns a data frame containing the observations of the resample.

outlier *Outlier history of single observation*

Description

`outlier` takes a `"robust2sls"` object and the index of a specific observation and returns its history of classification across the different iterations contained in the `"robust2sls"` object.

Usage

```
outlier(robust2sls_object, obs)
```

Arguments

robust2sls_object	An object of class `"robust2sls"`.
obs	An index (row number) of an observation

Value

`outlier` returns a vector that contains the 'type' value for the given observations across the different iterations. There are three possible values: 0 if the observations is judged to be an outlier, 1 if not, and -1 if any of its x, y, or z values required for estimation is missing.

outliers *Number of outliers*

Description

outliers calculates the number of outliers from a "robust2sls" object for a given iteration.

Usage

```
outliers(robust2sls_object, iteration)
```

Arguments

robust2sls_object	An object of class "robust2sls".
iteration	An integer >= 0 representing the iteration for which the outliers are calculated.

Value

outliers returns the number of outliers for a given iteration as determined by the outlier-detection algorithm.

outliers_prop *Proportion of outliers*

Description

outliers_prop calculates the proportion of outliers relative to all non-missing observations in the full sample from a "robust2sls" object for a given iteration.

Usage

```
outliers_prop(robust2sls_object, iteration)
```

Arguments

robust2sls_object	An object of class "robust2sls".
iteration	An integer >= 0 representing the iteration for which the outliers are calculated.

Value

outliers_prop returns the proportion of outliers for a given iteration as determined by the outlier-detection algorithm.

`outlier_detection` *Outlier detection algorithms*

Description

`outlier_detection` provides different types of outlier detection algorithms depending on the arguments provided. The decision whether to classify an observations as an outlier or not is based on its standardised residual in comparison to some user-specified reference distribution.
The algorithms differ mainly in two ways. First, they can differ by the use of initial estimator, i.e. the estimator based on which the first classification as outliers is made. Second, the algorithm can either be iterated a fixed number of times or until the difference in coefficient estimates between the most recent model and the previous one is smaller than some user-specified convergence criterion. The difference is measured by the L2 norm.

Usage

```
outlier_detection(
  data,
  formula,
  ref_dist = c("normal"),
  sign_level,
  initial_est = c("robustified", "saturated", "user", "iis"),
  user_model = NULL,
  iterations = 1,
  convergence_criterion = NULL,
  max_iter = NULL,
  shuffle = FALSE,
  shuffle_seed = NULL,
  split = 0.5,
  verbose = FALSE,
  iis_args = NULL
)
```

Arguments

`data`	A dataframe.
`formula`	A formula for the `ivreg` function, i.e. in the format `y ~ x1 + x2 | x1 + z2` where `y` is the dependent variable, `x1` are the exogenous regressors, `x2` the endogenous regressors, and `z2` the outside instruments.
`ref_dist`	A character vector that specifies the reference distribution against which observations are classified as outliers. `"normal"` refers to the normal distribution.
`sign_level`	A numeric value between 0 and 1 that determines the cutoff in the reference distribution against which observations are judged as outliers or not.
`initial_est`	A character vector that specifies the initial estimator for the outlier detection algorithm. `"robustified"` means that the full sample 2SLS is used as initial estimator. `"saturated"` splits the sample into two parts and estimates a 2SLS on each subsample. The coefficients of one subsample are used to calculate residuals and determine outliers in the other subsample. `"user"` allows the user to specify a model based on which observations are classified as outliers. `"iis"` applies impulse indicator saturation (IIS) as implemented in `ivisat`. See section "Warning" for more information and conditions.

`user_model` A model object of class ivreg. Only required if argument `initial_est` is set to `"user"`, otherwise `NULL`.

`iterations` Either an integer >= 0 that specifies how often the outlier detection algorithm is iterated, or the character vector `"convergence"`. In the former case, the value `0` means that only outlier classification based on the initial estimator is done. In the latter, the algorithm is iterated until it converges, i.e. when the difference in coefficient estimates between the most recent model and the previous one is smaller than some user-specified convergence criterion.

`convergence_criterion` A numeric value or NULL. The algorithm stops as soon as the difference in coefficient estimates between the most recent model and the previous one is smaller than `convergence_criterion`. The difference is measured by the L2 norm. If the argument is set to a numeric value but `iterations` is an integer > 0 then the algorithm stops either when it converged or when `iterations` is reached.

`max_iter` A numeric value >= 1 or NULL. If `iterations = "convergence"` is chosen, then the algorithm is stopped after at most `max_iter` iterations. If also a `convergence_criterion` is chosen then the algorithm stops when either the criterion is fulfilled or the maximum number of iterations is reached.

`shuffle` A logical value or `NULL`. Only used if `initial_est == "saturated"`. If `TRUE` then the sample is shuffled before creating the subsamples.

`shuffle_seed` An integer value that will set the seed for shuffling the sample or `NULL`. Only used if `initial_est == "saturated"` and `shuffle == TRUE`.

`split` A numeric value strictly between 0 and 1 that determines in which proportions the sample will be split.

`verbose` A logical value whether progress during estimation should be reported.

`iis_args` A list with named entries corresponding to the arguments for `iis_init` (`t.pval`, `do.pet`, `normality.JarqueB`, `turbo`, `overid`, `weak`). Can be `NULL` if `initial_est != "iis"`.

Value

`outlier_detection` returns an object of class `"robust2sls"`, which is a list with the following components:

`$cons` A list which stores high-level information about the function call and some results. `$call` is the captured function call, `$formula` the formula argument, `$data` the original data set, `$reference` the chosen reference distribution to classify outliers, `$sign_level` the significance level, `$psi` the probability that an observation is not classified as an outlier under the null hypothesis of no outliers, `$cutoff` the cutoff used to classify outliers if their standardised residuals are larger than that value, `$bias_corr` a bias correction factor to account for potential false positives (observations classified as outliers even though they are not). There are three further elements that are lists themselves.

`$initial` stores settings about the initial estimator: `$estimator` is the type of the initial estimator (e.g. robustified or saturated), `$split` how the sample is split (`NULL` if argument not used), `$shuffle` whether the sample is shuffled before splitting (`NULL` if argument not used), `$shuffle_seed` the value of the random seed (`NULL` if argument not used).

`$convergence` stores information about the convergence of the outlier-detection algorithm: `$criterion` is the user-specified convergence criterion (`NULL` if argument not used), `$difference`

is the L2 norm between the last coefficient estimates and the previous ones (NULL if argument not used or only initial estimator calculated). $converged is a logical value indicating whether the algorithm has converged, i.e. whether the difference is smaller than the convergence criterion (NULL if argument not used). $max_iter is the maximum iteration set by the user (NULL if argument not used or not set).

$iterations contains information about the user-specified iterations argument ($setting) and the actual number of iterations that were done ($actual). The actual number can be lower if the algorithm converged already before the user-specified number of iterations were reached.

$model A list storing the model objects of class ivreg for each iteration. Each model is stored under $m0, $m1, ...

$res A list storing the residuals of all observations for each iteration. Residuals of observations where any of the y, x, or z variables used in the 2SLS model are missing are set to NA. Each vector is stored under $m0, $m1, ...

$stdres A list storing the standardised residuals of all observations for each iteration. Standardised residuals of observations where any of the y, x, or z variables used in the 2SLS model are missing are set to NA. Standardisation is done by dividing by sigma, which is not adjusted for degrees of freedom. Each vector is stored under $m0, $m1, ...

$sel A list of logical vectors storing whether an observation is included in the estimation or not. Observations are excluded (FALSE) if they either have missing values in any of the x, y, or z variables needed in the model or when they are classified as outliers based on the model. Each vector is stored under $m0, $m1, ...

$type A list of integer vectors indicating whether an observation has any missing values in x, y, or z (-1), whether it is classified as an outlier (0) or not (1). Each vector is stored under $m0, $m1, ...

Warning

Check Jiao (2019) (as well as forthcoming working paper in the future) about conditions on the initial estimator that should be satisfied for the initial estimator when using initial_est == "user" (e.g. they have to be Op(1)). IIS is a generalisation of Saturated 2SLS with multiple block search but no asymptotic theory exists for IIS.

---

plot.robust2sls *Plotting of standardised residuals and outliers*

Description

Plot method for objects of class "robust2sls". Plots the standardised residuals of non-missing observations for a given iteration of the outlier-detection algorithm and distinguishes whether an observation is classified as an outlier by colour.

Usage

```
## S3 method for class 'robust2sls'
plot(x, iteration = NULL, ...)
```

Arguments

x	An object of class "robust2sls".
iteration	Either NULL (default) or an integer specifying the iteration that should be plotted. The default uses the final model.
...	Arguments to be passed to methods, see plot.

Value

plot.robust2sls returns a graph of class ggplot.

---

print.robust2sls *Helper of robust2sls class*

Description

robust2sls allows the user to create an object of class "robust2sls" by specifying the different components of the list. The validator function validate_robust2sls is called at the end to ensure that the resulting object is a valid object of class "robust2sls".

Usage

```
## S3 method for class 'robust2sls'
print(x, verbose = FALSE, ...)
```

Arguments

x	An object of class "robust2sls".
verbose	A logical value, TRUE or FALSE, determining whether detailed (TRUE) or shortened (FALSE) should be printed.
...	Further arguments passed to or from other methods, see print.

Details

Printing summary output

Print method for objects of class "robust2sls". Prints a high-level summary of the settings and results of the outlier-detection algorithm.

Value

No return value, prints model summary.

proptest *Proportion test*

Description

proptest() conducts a test whether the false outlier detection rate (FODR) in the sample deviates significantly from its expected value (population FODR) under the null hypothesis that there are no outliers in the sample.

Usage

```
proptest(robust2sls_object, alpha, iteration, one_sided = FALSE)
```

Arguments

robust2sls_object	An object of class "robust2sls" or a list of such objects.
alpha	A numeric value between 0 and 1 representing the significance level of the test.
iteration	An integer >= 0 or the character "convergence" that determines which iteration is used for the test.
one_sided	A logical value whether a two-sided test (FALSE) should be conducted or a one-sided test (TRUE) that rejects only when the false outlier detection rate is above its expected value.

Details

See outlier_detection() and multi_cutoff() for creating an object of class "robust2sls" or a list thereof.

Value

proptest() returns a data frame with the iteration (m) to be tested, the actual iteration that was tested (generally coincides with the iteration that was specified to be tested but is the convergent iteration if the fixed point is tested), the setting of the probability of exceeding the cut-off (gamma), the type of t-test (one- or two-sided), the value of the test statistic, its p-value, the significance level alpha, and the decision. The number of rows of the data frame corresponds to the length of the argument robust2sls_object.

robustified_init *Robustified 2SLS (full sample initial estimator)*

Description

robustified_init estimates the full sample 2SLS model, which is used as the initial estimator for the iterative procedure.

Usage

```
robustified_init(data, formula, cutoff)
```

Arguments

data	A dataframe.
formula	A formula in the format y ~ x1 + x2 | x1 + z2 where y is the dependent variable, x1 are the exogenous regressors, x2 the endogenous regressors, and z2 the outside instruments.
cutoff	A numeric cutoff value used to judge whether an observation is an outlier or not. If its absolute value is larger than the cutoff value, the observations is classified as an outlier.

Value

robustified_init returns a list with five elements. The first four are vectors whose length equals the number of observations in the data set. Unlike the residuals stored in a model object (usually accessible via model$residuals), it does not ignore observations where any of y, x or z are missing. It instead sets their values to NA.

The first element is a double vector containing the residuals for each observation based on the model estimates. The second element contains the standardised residuals, the third one a logical vector with TRUE if the observation is judged as not outlying, FALSE if it is an outlier, and NA if any of y, x, or z are missing. The fourth element of the list is an integer vector with three values: 0 if the observations is judged to be an outlier, 1 if not, and -1 if missing. The fifth and last element stores the ivreg model object based on which the four vectors were calculated.

---

saturated_init *Saturated 2SLS (split-sample initial estimator)*

Description

saturated_init splits the sample into two sub-samples. The 2SLS model is estimated on both sub-samples and the estimates of one sub-sample are used to calculate the residuals and hence outliers from the other sub-sample.

Usage

```
saturated_init(data, formula, cutoff, shuffle, shuffle_seed, split = 0.5)
```

Arguments

data	A dataframe.
formula	A formula in the format y ~ x1 + x2 | x1 + z2 where y is the dependent variable, x1 are the exogenous regressors, x2 the endogenous regressors, and z2 the outside instruments.
cutoff	A numeric cutoff value used to judge whether an observation is an outlier or not. If its absolute value is larger than the cutoff value, the observations is classified as an outlier.
shuffle	A logical value (TRUE or FALSE) whether the sample should be split into sub-samples randomly. If FALSE, the sample is simply cut into two parts using the original order of the supplied data set.
shuffle_seed	A numeric value that sets the seed for shuffling the data set before splitting it. Only used if shuffle == TRUE.
split	A numeric value strictly between 0 and 1 that determines in which proportions the sample will be split.

Value

saturated_init returns a list with five elements. The first four are vectors whose length equals the number of observations in the data set. Unlike the residuals stored in a model object (usually accessible via model$residuals), it does not ignore observations where any of y, x or z are missing. It instead sets their values to NA.

The first element is a double vector containing the residuals for each observation based on the model estimates. The second element contains the standardised residuals, the third one a logical vector with TRUE if the observation is judged as not outlying, FALSE if it is an outlier, and NA if any of y, x, or z are missing. The fourth element of the list is an integer vector with three values: 0 if the observations is judged to be an outlier, 1 if not, and -1 if missing. The fifth and last element is a list with the two initial ivreg model objects based on the two different sub-samples.

Warning

The estimator may have bad properties if the split is too unequal and the sample size is not large enough.

---

selection_iis *Create selection (non-outlying) vector from IIS model*

Description

selection_iis uses the data and isat model object to create a list with five elements that are used to determine whether the observations are judged as outliers or not.

Usage

```
selection_iis(x, data, yvar, complete, rownames_orig, refmodel)
```

Arguments

x	An object of class ivisat.
data	A dataframe.
yvar	A character vector of length 1 that refers to the name of the dependent variable in the data set.
complete	A logical vector with the same length as the number of observations in the data set that specifies whether an observation has any missing values in any of y, x, or z variables.
rownames_orig	A character vector storing the original rownames of the dataframe.
refmodel	A model object that will be stored in $model.

Value

A list with five elements. The first four are vectors whose length equals the number of observations in the data set. Unlike the residuals stored in a model object (usually accessible via model$residuals), it does not ignore observations where any of y, x or z are missing. It instead sets their values to NA.

The first element is a double vector containing the residuals for each observation based on the model estimates. The second element contains the standardised residuals, the third one a logical vector with TRUE if the observation is judged as not outlying, FALSE if it is an outlier, and NA if any

of y, x, or z are missing. The fourth element of the list is an integer vector with three values: 0 if the observations is judged to be an outlier, 1 if not, and -1 if missing. The fifth and last element stores the `ivreg` model object based on which the four vectors were calculated.

Note

IIS runs multiple models, similar to `saturated_init` but with multiple block search. These intermediate models are not recorded. For simplicity, the element `$model` of the returned list stores the full sample model result, identical to `robustified_init`.

Warning

Unlike the residuals stored in a model object (usually accessible via `model$residuals`), this function returns vectors of the same length as the original data set even if any of the y, x, or z variables are missing. The residuals for those observations are set to `NA`.

`sumtest` *Scaling sum proportion test across different cut-offs*

Description

`sumtest()` uses the estimations across several cut-offs to test whether the sum of the deviations between sample and population FODR differ significantly from its expected value.

$$\sum_{k=1}^{K} \sqrt{n}(\widehat{\gamma}_{c_k} - \gamma_{c_k})$$

Usage

```
sumtest(robust2sls_object, alpha, iteration, one_sided = FALSE)
```

Arguments

`robust2sls_object`	A list of `"robust2sls"` objects.
`alpha`	A numeric value between 0 and 1 representing the significance level of the test.
`iteration`	An integer >= 0 or the character "convergence" that determines which iteration is used for the test.
`one_sided`	A logical value whether a two-sided test (`FALSE`) should be conducted or a one-sided test (`TRUE`) that rejects only when the false outlier detection rate is above its expected value.

Value

`sumtest()` returns a data frame with one row storing the iteration that was tested, the value of the test statistic (t-test), the type of the test (one- or two-sided), the corresponding p-value, the significance level, and whether the null hypothesis is rejected. The data frame also contains an attribute named `"gammas"` that records which gammas determining the different cut-offs were used in the scaling sum test.

suptest *Supremum proportion test across different cut-offs*

Description

suptest() uses the estimations across several cut-offs to test whether the supremum/maximum of the deviations between sample and population FODR differs significantly from its expected value.

$$\sup_c |\sqrt{n}(\widehat{\gamma}_c - \gamma_c)|$$

Usage

```
suptest(robust2sls_object, alpha, iteration, p = c(0.9, 0.95, 0.99), R = 50000)
```

Arguments

robust2sls_object	A list of "robust2sls" objects.
alpha	A numeric value between 0 and 1 representing the significance level of the test.
iteration	An integer >= 0 or the character "convergence" that determines which iteration is used for the test.
p	A numeric vector of probabilities with values in [0,1] for which the corresponding quantiles are calculated.
R	An integer specifying the number of replications for simulating the distribution of the test statistic.

Value

suptest() returns a data frame with one row storing the iteration that was tested, the value of the test statistic, the corresponding p-value, the significance level, and whether the null hypothesis is rejected. The data frame also contains two named attributes. The first attribute is named "gammas" and records which gammas determining the different cut-offs were used in the scaling sup test. The second attribute is named "critical" and records the critical values corresponding to the different quantiles in the limiting distribution that were specified in p.

user_init *User-specified initial estimator*

Description

user_init uses a model supplied by the user as the initial estimator. Based on this estimator, observations are classified as outliers or not.

Usage

```
user_init(data, formula, cutoff, user_model)
```

Arguments

data	A dataframe.
formula	A formula in the format y ~ x1 + x2 | x1 + z2 where y is the dependent variable, x1 are the exogenous regressors, x2 the endogenous regressors, and z2 the outside instruments.
cutoff	A numeric cutoff value used to judge whether an observation is an outlier or not. If its absolute value is larger than the cutoff value, the observations is classified as an outlier.
user_model	A model object of class ivreg whose parameters are used to calculate the residuals.

Value

user_init returns a list with five elements. The first four are vectors whose length equals the number of observations in the data set. Unlike the residuals stored in a model object (usually accessible via model$residuals), it does not ignore observations where any of y, x or z are missing. It instead sets their values to NA.

The first element is a double vector containing the residuals for each observation based on the model estimates. The second element contains the standardised residuals, the third one a logical vector with TRUE if the observation is judged as not outlying, FALSE if it is an outlier, and NA if any of y, x, or z are missing. The fourth element of the list is an integer vector with three values: 0 if the observations is judged to be an outlier, 1 if not, and -1 if missing. The fifth and last element stores the ivreg user-specified model object based on which the four vectors were calculated.

Warning

Check Jiao (2019) about conditions on the initial estimator that should be satisfied for the initial estimator (e.g. they have to be Op(1)).

---

validate_robust2sls *Validator of robust2sls class*

Description

validate_robust2sls checks that the input is a valid object of class "robust2sls".

Usage

```
validate_robust2sls(x)
```

Arguments

x	An object whose validity of class "robust2sls" is tested.

Value

If the object is a valid "robust2sls" object then the function returns the object. No return value otherwise.

varrho *Calculate varrho coefficients*

Description

varrho calculates the coefficients for the asymptotic variance of the gauge (false outlier detection rate) for a specific iteration m >= 1.

Usage

```
varrho(sign_level, ref_dist = c("normal"), iteration)
```

Arguments

sign_level	A numeric value between 0 and 1 that determines the cutoff in the reference distribution against which observations are judged as outliers or not.
ref_dist	A character vector that specifies the reference distribution against which observations are classified as outliers. "normal" refers to the normal distribution.
iteration	An integer >= 1 that specifies the iteration of the outlier detection algorithm.

Value

varrho returns a list with four components, all of which are lists themselves. $setting stores the arguments with which the function was called. $c stores the values of the six different coefficients for the specified iteration. $fp contains the fixed point versions of the six coefficients. $aux stores intermediate values required for calculating the coefficients.

Index