Package ‘fwildclusterboot’

June 10, 2022

Title Fast Wild Cluster Bootstrap Inference for Linear Models
Version 0.9
Date 2022-06-06

Description Implementation of the fast algorithm for wild cluster bootstrap inference developed in Roodman et al (2019, STATA Journal) for linear regression models <doi:10.1177/1536867X19830877>, which makes it feasible to quickly calculate bootstrap test statistics based on a large number of bootstrap draws even for large samples. Multiway clustering, regression weights, bootstrap weights, fixed effects and subcluster bootstrapping are supported. Further, both restricted (WCR) and unrestricted (WCU) bootstrap are supported. Methods are provided for a variety of fitted models, including 'lm()', 'feols()' (from package 'fixest') and 'felm()' (from package 'lfe'). Additionally implements a heteroskedasticity-robust (HC1) wild bootstrap.

Further, the package provides an R binding to 'WildBootTests.jl', which provides additional speed gains and functionality, including the 'WRE' bootstrap for instrumental variable models (based on models of type 'ivreg()' from package 'ivreg') and hypotheses with q > 1.

URL https://s3alfisc.github.io/fwildclusterboot/

BugReports https://github.com/s3alfisc/fwildclusterboot/issues/

License GPL-3

Imports collapse, Formula, Rcpp, dreamerr, Matrix, Matrix.utils, generics, gtools, dqrng, JuliaConnectoR

Suggests fixest, lfe, ivreg, clubSandwich, sandwich, lmtest, data.table, fabricatr, covr, knitr, rmarkdown, broom, modelsummary, bench, testthat (>= 3.0.0), tibble

Encoding UTF-8

LazyData true

RoxygenNote 7.2.0
### R topics documented:

- `.onLoad` .................................................. 3
- `boottest` ...................................................... 3
- `boottest.felm` ................................................ 4
- `boottest.fixest` ............................................ 9
- `boottest.ivreg` ............................................. 15
- `boottest.lm` ............................................... 19
- `boot_algo1` .................................................. 23
- `boot_algo2` .................................................. 25
- `boot_ssc` ..................................................... 26
- `cpp_get_nb_threads` ....................................... 27
- `create_data` ............................................... 27
- `eigenMapMatMult` .......................................... 28
- `get_bootstrap_pvalue` .................................... 29
- `get_seed` ..................................................... 29
- `get_ssc` ..................................................... 30
- `glance.boottest` ........................................... 30
- `mboottest` .................................................. 31
- `mboottest.felm` ............................................ 32
- `mboottest.fixest` ........................................ 35
- `mboottest.lm` .............................................. 39
- `model_matrix` ............................................... 42
- `model_matrix.felm` ....................................... 43
- `model_matrix.fixest` .................................... 43
- `model_matrix.lm` ......................................... 44
- `plot.boottest` .............................................. 44
- `preprocess2` ............................................... 45
.onLoad

setting options for nthreads when package is loaded

Description

setting options for nthreads when package is loaded

Usage

.onLoad(libname, pkgname)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libname</td>
<td>library name</td>
</tr>
<tr>
<td>pkgname</td>
<td>package name</td>
</tr>
</tbody>
</table>

Value

Changes number of threads used.

boottest  Fast wild cluster bootstrap inference

Description

boottest is a S3 method that allows for fast wild cluster bootstrap inference for objects of class lm, fixest and felm by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019.

Usage

boottest(object, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object of type lm, fixest, felm or ivreg</td>
</tr>
<tr>
<td>...</td>
<td>other arguments</td>
</tr>
</tbody>
</table>
Value

An object of class boottest.

Setting Seeds

To guarantee reproducibility, you can either use boottest()’s seed function argument, or set a global random seed via

- `set.seed()` when using
  1. the lean algorithm (via `boot_algo = "R-lean"`), 2) the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via `boot_algo = "R"` with Mammen weights or
  4) boot_algo = "WildBootTests.jl"
- `dqrng::dqset.seed()` when using `boot_algo = "R"` for Rademacher, Webb or Normal weights

References


See Also

boottest.lm, boottest.fixest, boottest.felm, boottest.ivreg

boottest.felm Fast wild cluster bootstrap inference for object of class felm

Description

boottest.felm is a S3 method that allows for fast wild cluster bootstrap inference for objects of class felm by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019 and implemented in the STATA package boottest.
boottest.felm

Usage

```r
## S3 method for class 'felm'
boottest(
  object,
  param,
  B,
  clustid = NULL,
  bootcluster = "max",
  fe = NULL,
  conf_int = TRUE,
  seed = NULL,
  R = NULL,
  r = 0,
  beta0 = NULL,
  sign_level = 0.05,
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  maxiter = 10,
  nthreads = getBoottest_nthreads(),
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df = "conventional"),
  boot_algo = getBoottest_boot_algo(),
  floattype = "Float64",
  maxmatsize = FALSE,
  bootstrapc = FALSE,
  t_boot = FALSE,
  getauxweights = FALSE,
  ...
)
```

Arguments

- **object**: An object of class felm
- **param**: A character vector or rhs formula. The name of the regression coefficient(s) for which the hypothesis is to be tested
- **B**: Integer. The number of bootstrap iterations. When the number of clusters is low, increasing B adds little additional runtime.
- **clustid**: A character vector or rhs formula containing the names of the cluster variables. If NULL, a heteroskedasticity-robust (HC1) wild bootstrap is run.
- **bootcluster**: A character vector or rhs formula of length 1. Specifies the bootstrap clustering variable or variables. If more than one variable is specified, then bootstrapping is clustered by the intersections of clustering implied by the listed variables. To mimic the behavior of stata's boottest command, the default is to cluster by the intersection of all the variables specified via the clustid argument, even though that is not necessarily recommended (see the paper by...
Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.

\textbf{fe}  
A character vector or rhs formula of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, fe needs to be NULL.

\textbf{conf_int}  
A logical vector. If TRUE, boottest computes confidence intervals by test inversion. If FALSE, only the p-value is returned.

\textbf{seed}  
An integer. Allows to set a random seed. For details, see below.

\textbf{R}  
Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as \texttt{param}. If NULL, a vector of ones of length \texttt{param}.

\textbf{r}  
A numeric. Shifts the null hypothesis H0: param = r vs H1: param != r

\textbf{beta0}  
Deprecated function argument. Replaced by function argument ‘r’.

\textbf{sign_level}  
A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. sign_level = 0.05 returns 95% confidence intervals. By default, sign_level = 0.05.

\textbf{type}  
character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher distribution, if the number of replications B exceeds the number of possible draw combinations, $2^{(#\text{number of clusters})}$, then boottest() will use each possible combination once (enumeration).

\textbf{impose_null}  
Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)

\textbf{p_val_type}  
Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".

\textbf{tol}  
Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. 1e-6 by default.

\textbf{maxiter}  
Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.

\textbf{nthreads}  
The number of threads. Can be: a) an integer lower than, or equal to, the maximum number of threads; b) 0: meaning all available threads will be used; c) a number strictly between 0 and 1 which represents the fraction of all threads to use. The default is to use 1 core.

\textbf{ssc}  
An object of class boot_ssc.type obtained with the function boot_ssc. Represents how the small sample adjustments are computed. The defaults are \texttt{adj = TRUE}, fixef.K = "none", You can find more details in the help file for boot_ssc(). The function is purposefully designed to mimic fixest’s ssc function.

\textbf{boot_algo}  
Character scalar. Either "R" or "WildBootTests.jl". Controls the algorithm employed by boottest. "R" is the default and implements the cluster bootstrap as in Roodman (2019). "WildBootTests.jl" executes the wild cluster bootstrap by via the WildBootTests.jl package. For it to run, Julia and WildBootTests.jl need to
be installed. Check out the set_up_... functions The "fast and wild" algorithm is extremely fast for small number of clusters, but because it is fully vectorized, very memory-demanding. For large number of clusters and large number of bootstrap iterations, the fast and wild algorithm becomes infeasible. If a out-of-memory error # occurs, the "lean" algorithm is a memory friendly, but less performant rcpp-armadillo based implementation of the wild cluster bootstrap. Note that if no cluster is provided, boottest() always defaults to the "lean" algorithm. Note that you can set the employed algorithm globally by using the setBoottest_boot_algo() function.

**floattype** Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit? Only relevant when 'boot_algo = "WildBootTests.jl"

**maxmatsize** NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes. Only relevant when 'boot_algo = "WildBootTests.jl"

**bootstrapc** Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t. Only relevant when 'boot_algo = "WildBootTests.jl"

**t_boot** Logical. Should bootstrapped t-statistics be returned?

**getauxweights** Logical. Whether to save auxilliary weight matrix (v)

... Further arguments passed to or from other methods.

**Value**

An object of class boottest

- **p_val** The bootstrap p-value.
- **conf_int** The bootstrap confidence interval.
- **param** The tested parameter.
- **N** Sample size. Might differ from the regression sample size if the cluster variables contain NA values.
- **boot_iter** Number of Bootstrap Iterations.
- **clustid** Names of the cluster Variables.
- **N_G** Dimension of the cluster variables as used in boottest.
- **sign_level** Significance level used in boottest.
- **type** Distribution of the bootstrap weights.
- **impose_null** Whether the null was imposed on the bootstrap dgp or not.
- **R** The vector "R" in the null hypothesis of interest Rbeta = r.
- **r** The scalar "r" in the null hypothesis of interest Rbeta = r.
- **point_estimate** R’beta. A scalar: the constraints vector times the regression coefficients.
- **grid_vals** All t-statistics calculated while calculating the confidence interval.
- **p_grid_vals** All p-values calculated while calculating the confidence interval.
- **t_stat** The 'original' regression test statistics.
Setting Seeds

To guarantee reproducibility, you can either use boottest()'s seed function argument, or set a global random seed via

- `set.seed()` when using
  1. the lean algorithm (via `boot_algo = "R-lean"`) including the heteroskedastic wild boot-strap
  2. the wild cluster bootstrap via `boot_algo = "R"` with Mammen weights or
  3. `boot_algo = "WildBootTests.jl"
- `dqrng::dqset.seed()` when using `boot_algo = "R"` for Rademacher, Webb or Normal weights

Confidence Intervals

boottest computes confidence intervals by inverting p-values. In practice, the following procedure is used:

- Based on an initial guess for starting values, calculate p-values for 26 equal spaced points between the starting values.
- Out of the 26 calculated p-values, find the two pairs of values x for which the corresponding p-values px cross the significance level sign_level.
- Feed the two pairs of x into an numerical root finding procedure and solve for the root. boottest currently relies on `stats::uniroot` and sets an absolute tolerance of 1e-06 and stops the procedure after 10 iterations.

Standard Errors

boottest does not calculate standard errors.

References

Examples

```r
## Not run:
if (requireNamespace("lfe")) {
  library(lfe)
  data(voters)
  felm_fit <- felm(proposition_vote ~ treatment + ideology1 + log_income |  
                   Q1_immigration,
                   data = voters)
  boot1 <- boottest(felm_fit,
                    B = 9999,
                    param = "treatment",
                    clustid = "group_id1"
                  )
  boot2 <- boottest(felm_fit,
                    B = 9999,
                    param = "treatment",
                    clustid = c("group_id1", "group_id2")
                  )
  boot3 <- boottest(felm_fit,
                    B = 9999,
                    param = "treatment",
                    clustid = c("group_id1", "group_id2"),
                    fe = "Q1_immigration"
                  )
  boot4 <- boottest(felm_fit,
                    B = 999,
                    param = "treatment",
                    clustid = c("group_id1", "group_id2"),
                    fe = "Q1_immigration",
                    sign_level = 0.2,
                    seed = 8,
                    r = 2
                  )
  # test treatment + ideology1 = 2
  boot5 <- boottest(felm_fit,
                    B = 9999,
                    clustid = c("group_id1", "group_id2"),
                    param = c("treatment", "ideology1"),
                    R = c(1, 1),
                    r = 2
                  )
  summary(boot1)
  plot(boot1)
}
## End(Not run)
```

boottest.fixest

*Fast wild cluster bootstrap inference for object of class fixest*
boottest.fixest is a S3 method that allows for fast wild cluster bootstrap inference for objects of class fixest by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019 and implemented in the STATA package boottest.

Usage

```r
## S3 method for class 'fixest'
boottest(
oobject, 
param, 
B, 
clustid = NULL, 
bootcluster = "max", 
fe = NULL, 
sign_level = 0.05, 
conf_int = TRUE, 
seed = NULL, 
R = NULL, 
r = 0, 
beta0 = NULL, 
type = "rademacher", 
impose_null = TRUE, 
p_val_type = "two-tailed", 
tol = 1e-06, 
maxiter = 10, 
nthreads = getBoottest_nthreads(), 
ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df = "conventional"), 
boot_algo = getBoottest_boot_algo(), 
floattype = "Float64", 
maxmatsize = FALSE, 
bootstrapc = FALSE, 
t_boot = FALSE, 
getauxweights = FALSE, 
... 
)
```

Arguments

- `object`: An object of class fixest and estimated via `fixest::feols()`. Non-linear models are not supported.
- `param`: A character vector or rhs formula. The name of the regression coefficient(s) for which the hypothesis is to be tested.
- `B`: Integer. The number of bootstrap iterations. When the number of clusters is low, increasing B adds little additional runtime.
- `clustid`: A character vector or rhs formula containing the names of the cluster variables. If NULL, a heteroskedasticity-robust (HC1) wild bootstrap is run.
boottest.fixest

bootcluster  A character vector or rhs formula of length 1. Specifies the bootstrap clustering variable or variables. If more than one variable is specified, then bootstrapping is clustered by the intersections of clustering implied by the listed variables. To mimic the behavior of stata’s boottest command, the default is to cluster by the intersection of all the variables specified via the clustid argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.

fe  A character vector or rhs formula of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, fe needs to be NULL.

sign_level  A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. sign_level = 0.05 returns 0.95% confidence intervals. By default, sign_level = 0.05.

cnf_int  A logical vector. If TRUE, boottest computes confidence intervals by test inversion. If FALSE, only the p-value is returned.

seed  An integer. Allows to set a random seed. For details, see below.

R  Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as param. If NULL, a vector of ones of length param.

r  A numeric. Shifts the null hypothesis H0: param = r vs H1: param != r

beta0  Deprecated function argument. Replaced by function argument ‘r’.

type  character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher distribution, if the number of replications B exceeds the number of possible draw ombinations, 2^(#number of clusters), then boottest() will use each possible combination once (enumeration).

impose_null  Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)

p val_type  Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".

tol  Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. 1e-6 by default.

maxiter  Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.

nthreads  The number of threads. Can be: a) an integer lower than, or equal to, the maximum number of threads; b) 0: meaning all available threads will be used; c) a number strictly between 0 and 1 which represents the fraction of all threads to use. The default is to use 1 core.

ssc  An object of class boot_ssc.type obtained with the function boot_ssc. Represents how the small sample adjustments are computed. The defaults are adj = TRUE, fixef.K = "none", You can find more details in the help file for boot_ssc(). The function is purposefully designed to mimic fixest’s ssc function.
boot.test.fixest

boot.algo Character scalar. Either "R" or "WildBootTests.jl". Controls the algorithm employed by boottest(). "R" is the default and implements the cluster bootstrap as in Roodman (2019). "WildBootTests.jl" executes the wild cluster bootstrap via the WildBootTests.jl package. For it to run, Julia and WildBootTests.jl need to be installed. Note that if no cluster is provided, boottest() always defaults to the "lean" algorithm. You can set the employed algorithm globally by using the setBoottest_boot.algo() function.

floattype Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit? Only relevant when 'boot.algo = "WildBootTests.jl"'

maxmatsize NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes. Only relevant when 'boot.algo = "WildBootTests.jl"'

bootstrapc Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t. Only relevant when 'boot.algo = "WildBootTests.jl"'

t.boot Logical. Should bootstrapped t-statistics be returned?

getauxweights Logical. Whether to save auxilliary weight matrix (v)

Value

An object of class boottest

p.val The bootstrap p-value.

conf.int The bootstrap confidence interval.

param The tested parameter.

N Sample size. Might differ from the regression sample size if the cluster variables contain NA values.

boot.iter Number of Bootstrap Iterations.

clusid Names of the cluster Variables.

N_G Dimension of the cluster variables as used in boottest.

sign_level Significance level used in boottest.

type Distribution of the bootstrap weights.

impose_null Whether the null was imposed on the bootstrap dgp or not.

R The vector "R" in the null hypothesis of interest Rbeta = r.

r The scalar 'r' in the null hypothesis of interest Rbeta = r.

point_estimate R'beta. A scalar: the constraints vector times the regression coefficients.

grid_vals All t-statistics calculated while calculating the confidence interval.

p.grid_vals All p-values calculated while calculating the confidence interval.

t.stat The ‘original’ regression test statistics.

t.boot All bootstrap t-statistics.

regression The regression object used in boottest.
**Setting Seeds**

To guarantee reproducibility, you can either use `boottest()`’s seed function argument, or set a global random seed via

- `set.seed()` when using
  1. the lean algorithm (via `boot_algo = "R-lean"`) including the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via `boot_algo = "R"` with Mammen weights or
  3. `boot_algo = "WildBootTests.jl"
- `dqrng::dqset.seed()` when using `boot_algo = "R"` for Rademacher, Webb or Normal weights

**Confidence Intervals**

`boottest` computes confidence intervals by inverting p-values. In practice, the following procedure is used:

- Based on an initial guess for starting values, calculate p-values for 26 equal spaced points between the starting values.
- Out of the 26 calculated p-values, find the two pairs of values x for which the corresponding p-values px cross the significance sign_level sign_level.
- Feed the two pairs of x into an numerical root finding procedure and solve for the root. `boottest` currently relies on `stats::uniroot` and sets an absolute tolerance of 1e-06 and stops the procedure after 10 iterations.

**Standard Errors**

`boottest` does not calculate standard errors.

**References**


Examples

## Not run:
if (requireNamespace("fixest")) {
  library(fwildclusterboot)
  library(fixest)
  data(voters)
  feols_fit <- feols(proposition_vote ~ treatment + ideology1 + log_income,
                     fixef = "Q1_immigration",
                     data = voters
       )
  boot1 <- boottest(feols_fit,
                     B = 9999,
                     param = "treatment",
                     clustid = "group_id1"
       )
  boot2 <- boottest(feols_fit,
                     B = 9999,
                     param = "treatment",
                     clustid = c("group_id1", "group_id2")
       )
  boot3 <- boottest(feols_fit,
                     B = 9999,
                     param = "treatment",
                     clustid = c("group_id1", "group_id2"),
                     fe = "Q1_immigration"
       )
  boot4 <- boottest(feols_fit,
                     B = 9999,
                     param = "treatment",
                     clustid = c("group_id1", "group_id2"),
                     fe = "Q1_immigration",
                     sign_level = 0.2,
                     seed = 8,
                     r = 2
       )
  # test treatment + ideology1 = 2
  boot5 <- boottest(feols_fit,
                     B = 9999,
                     clustid = c("group_id1", "group_id2"),
                     param = c("treatment", "ideology1"),
                     R = c(1, 1),
                     r = 2
       )
  summary(boot1)
  plot(boot1)
}

## End(Not run)
boottest.ivreg

**Fast wild cluster bootstrap inference for object of class lm**

**Description**

boottest.ivreg is a S3 method that allows for fast wild cluster bootstrap inference for objects of class ivreg by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019 for instrumental variable models (WRE, Davidson & McKinnon, 2010).

**Usage**

```r
## S3 method for class 'ivreg'
boottest(
  object, clustid, param, B, bootcluster = "max", conf_int = TRUE, seed = NULL, R = NULL, 
  r = 0, sign_level = 0.05, type = "rademacher", impose_null = TRUE, 
  p_val_type = "two-tailed", tol = 1e-06, floattype = "Float64", 
  getauxweights = FALSE, t_boot = FALSE, maxmatsize = NULL, 
  bootstrapc = FALSE, liml = FALSE, fuller = NULL, kappa = NULL, 
  arubin = FALSE, ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df = 
    "conventional"), ...
)
```

**Arguments**

- `object`: An object of class lm
- `clustid`: A character vector or rhs formula containing the names of the cluster variables
- `param`: A character vector or rhs formula of length one. The name of the regression coefficient for which the hypothesis is to be tested
\( B \) \hspace{1cm} \text{Integer. The number of bootstrap iterations. When the number of clusters is low, increasing \( B \) adds little additional runtime.}

\texttt{bootcluster} \hspace{1cm} \text{A character vector or rhs formula of length 1. Specifies the bootstrap clustering variable or variables. If more than one variable is specified, then bootstrapping is clustered by the intersections of clustering implied by the listed variables. To mimic the behavior of stata's \texttt{boottest} command, the default is to cluster by the intersection of all the variables specified via the \texttt{clustid} argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.}

\texttt{conf_int} \hspace{1cm} \text{A logical vector. If TRUE, \texttt{boottest} computes confidence intervals by test inversion. If FALSE, only the \( p \)-value is returned.}

\texttt{seed} \hspace{1cm} \text{An integer. Allows to set a random seed. For details, see below.}

\( R \) \hspace{1cm} \text{Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as \texttt{param}. If NULL, a vector of ones of length \texttt{param}.}

\( r \) \hspace{1cm} \text{A numeric. Shifts the null hypothesis \( H_0: \text{param} = r \) vs \( H_1: \text{param} \neq r \)}

\texttt{sign_level} \hspace{1cm} \text{A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. \texttt{sign_level} = 0.05 returns 0.95\% confidence intervals. By default, \texttt{sign_level} = 0.05.}

\texttt{type} \hspace{1cm} \text{character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively, \texttt{type} can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher and Mammen distribution, if the number of replications \( B \) exceeds the number of possible draw ombinations, \( 2^{(#\text{number of clusters})} \), then \texttt{boottest()} will use each possible combination once (enumeration).}

\texttt{impose_null} \hspace{1cm} \text{Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (\( \texttt{WCR} \)) by default. If \texttt{FALSE}, the null is not imposed (\( \texttt{WCU} \)).}

\texttt{p_val_type} \hspace{1cm} \text{Character vector of length 1. Type of \( p \)-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".}

\texttt{tol} \hspace{1cm} \text{Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. Relative tolerance of 1e-6 by default.}

\texttt{floattype} \hspace{1cm} \text{Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit?}

\texttt{getauxweights} \hspace{1cm} \text{Logical. \texttt{FALSE} by default. Whether to save auxilliary weight matrix (\( v \)).}

\texttt{t_boot} \hspace{1cm} \text{Logical. Should bootstrapped \( t \)-statistics be returned?}

\texttt{maxmatsize} \hspace{1cm} \text{NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (\( v \)), in gigabytes.}

\texttt{bootstrapc} \hspace{1cm} \text{Logical scalar, \texttt{FALSE} by default. \texttt{TRUE} to request bootstrap-c instead of bootstrap-t.
`boottest(ivreg)`

- **liml**: Logical scalar. False by default. TRUE for liml or fuller liml.
- **fuller**: NULL by default. Numeric scalar. fuller liml factor.
- **kappa**: Null by default. fixed $<\U+03BA>$ for k-class estimation.
- **arubin**: False by default. Logical scalar. TRUE for Anderson-Rubin Test.
- **ssc**: An object of class `boot_ssc.type` obtained with the function `boot_ssc`. Represents how the small sample adjustments are computed. The defaults are `adj = TRUE`, `fixef.K = "none"`, You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest’s `ssc` function. Further arguments passed to or from other methods.

**Value**

An object of class `boottest`

- **p_val**: The bootstrap p-value.
- **conf_int**: The bootstrap confidence interval.
- **param**: The tested parameter.
- **N**: Sample size. Might differ from the regression sample size if the cluster variables contain NA values.
- **boot_iter**: Number of Bootstrap Iterations.
- **clustid**: Names of the cluster Variables.
- **N_G**: Dimension of the cluster variables as used in boottest.
- **sign_level**: Significance level used in boottest.
- **type**: Distribution of the bootstrap weights.
- **impose_null**: Whether the null was imposed on the bootstrap dgp or not.
- **R**: The vector "$R$" in the null hypothesis of interest $R\beta = r$.
- **r**: The scalar "$r$" in the null hypothesis of interest $R\beta = r$.
- **point_estimate**: $R'\beta$. A scalar: the constraints vector times the regression coefficients.
- **grid_vals**: All t-statistics calculated while calculating the confidence interval.
- **p_grid_vals**: All p-values calculated while calculating the confidence interval.
- **t_stat**: The ’original’ regression test statistics.
- **t_boot**: All bootstrap t-statistics.
- **regression**: The regression object used in boottest.
- **call**: Function call of boottest.
- **boot_algo**: The employed bootstrap algorithm.
- **nthreads**: The number of threads employed.
- **internal_seed**: The integer value -inherited from set.seed() - used within boottest() to set the random seed in either R or Julia. If NULL, no internal seed was created.
Setting Seeds

To guarantee reproducibility, you can either use `boottest()`'s seed function argument, or set a global random seed via

- `set.seed()` when using
  1. the lean algorithm (via `boot_algo = "R-lean"`) including the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via `boot_algo = "R"` with Mammen weights or
  3. `boot_algo = "WildBootTests.jl"

- `dqrng::dqset.seed()` when using `boot_algo = "R"` for Rademacher, Webb or Normal weights

References


Examples

```r
## Not run:
library(ivreg)
library(fwildclusterboot)

# drop all NA values from SchoolingReturns
SchoolingReturns <- SchoolingReturns[rowMeans(sapply(SchoolingReturns, is.na)) == 0, ]
ivreg_fit <- ivreg(log(wage) ~ education + age +
                   ethnicity + smsa + south + parents14 |
                   nearcollege + age + ethnicity + smsa
                   + south + parents14,
                   data = SchoolingReturns
)

boot_ivreg <- boottest(
  object = ivreg_fit,
  B = 999,
  param = "education",
  clustid = "kww",
  type = "mammen",
  impose_null = TRUE
)
summary(boot_ivreg)
```
boottest.lm

Fast wild cluster bootstrap inference for object of class lm

Description

boottest.lm is a S3 method that allows for fast wild cluster bootstrap inference for objects of class lm by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019.

Usage

## S3 method for class 'lm'
boottest(
  object, param, B, clustid = NULL, bootcluster = "max", conf_int = TRUE, seed = NULL, R = NULL, r = 0, beta0 = NULL, sign_level = 0.05, type = "rademacher", impose_null = TRUE, p_val_type = "two-tailed", tol = 1e-06, maxiter = 10, nthreads = getBoottest_nthreads(), ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df = "conventional"), boot_algo = getBoottest_boot_algo(), floattype = "Float64", maxmatsize = FALSE, bootstrapc = FALSE, t_boot = FALSE, getauxweights = FALSE, ...
)

Arguments

object An object of class lm
param A character vector or rhs formula. The name of the regression coefficient(s) for which the hypothesis is to be tested.

B Integer. The number of bootstrap iterations. When the number of clusters is low, increasing B adds little additional runtime.

clustid A character vector or rhs formula containing the names of the cluster variables. If NULL, a heteroskedasticity-robust (HC1) wild bootstrap is run.

bootcluster A character vector or rhs formula of length 1. Specifies the bootstrap clustering variable or variables. If more than one variable is specified, bootstrapping is clustered by the intersections of clustering implied by the listed variables. To mimic the behavior of stata’s boottest command, the default is to cluster by the intersection of all the variables specified via the clustid argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include ”min”, where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette(“fwildclusterboot”, package = “fwildclusterboot”) for details.

conf_int A logical vector. If TRUE, boottest computes confidence intervals by test inversion. If FALSE, only the p-value is returned.

seed An integer. Allows to set a random seed. For details, see below.

R Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as param. If NULL, a vector of ones of length param.

r A numeric. Shifts the null hypothesis H0: param = r vs H1: param != r

beta0 Deprecated function argument. Replaced by function argument ‘r’.

sign_level A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. sign_level = 0.05 returns 0.95% confidence intervals. By default, sign_level = 0.05.

type character or function. The character string specifies the type of bootstrap to use: One of ”rademacher”, ”mammen”, ”norm” and ”webb”. Alternatively, type can be a function(n) for drawing wild bootstrap factors. ”rademacher” by default. For the Rademacher distribution, if the number of replications B exceeds the number of possible draw combinations, 2^(#number of clusters), then boottest() will use each possible combination once (enumeration).

impose_null Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)

p_val_type Character vector of length 1. Type of p-value. By default ”two-tailed”. Other options include ”equal-tailed”, “>” and “<”.

tol Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. 1e-6 by default.

maxiter Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.

nthreads The number of threads. Can be: a) an integer lower than, or equal to, the maximum number of threads; b) 0: meaning all available threads will be used; c) a number strictly between 0 and 1 which represents the fraction of all threads to use. The default is to use 1 core.
**Value**

An object of class `boottest`

- **p_val** The bootstrap p-value.
- **conf_int** The bootstrap confidence interval.
- **param** The tested parameter.
- **N** Sample size. Might differ from the regression sample size if the cluster variables contain NA values.
- **boot_iter** Number of Bootstrap Iterations.
- **clustid** Names of the cluster Variables.
- **N_G** Dimension of the cluster variables as used in boottest.
- **sign_level** Significance level used in boottest.
- **type** Distribution of the bootstrap weights.
- **impose_null** Whether the null was imposed on the bootstrap dgp or not.
- **R** The vector "R" in the null hypothesis of interest \( R\beta = r \).
- **r** The scalar "r" in the null hypothesis of interest \( R\beta = r \).
- **point_estimate** \( R'\beta \). A scalar: the constraints vector times the regression coefficients.
- **grid_vals** All t-statistics calculated while calculating the confidence interval.
p_grid_vals  All p-values calculated while calculating the confidence interval.
t_stat      The 'original' regression test statistics.
t_boot      All bootstrap t-statistics.
regression  The regression object used in boottest.
call        Function call of boottest.
boot_algo   The employed bootstrap algorithm.
nthreads    The number of threads employed.
internal_seed The integer value -inherited from set.seed() - used within boottest() to set the random seed in either R or Julia. If NULL, no internal seed was created.

Setting Seeds

To guarantee reproducibility, you can either use boottest()'s seed function argument, or set a global random seed via

- set.seed() when using
  1. the lean algorithm (via boot_algo = "R-lean") including the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via boot_algo = "R" with Mammen weights or
  3. boot_algo = "WildBootTests.jl"
- dqrng::dqset.seed() when using boot_algo = "R" for Rademacher, Webb or Normal weights

Confidence Intervals

boottest computes confidence intervals by inverting p-values. In practice, the following procedure is used:

- Based on an initial guess for starting values, calculate p-values for 26 equal spaced points between the starting values.
- Out of the 26 calculated p-values, find the two pairs of values x for which the corresponding p-values px cross the significance level sign_level.
- Feed the two pairs of x into an numerical root finding procedure and solve for the root. boottest currently relies on stats::uniroot and sets an absolute tolerance of 1e-06 and stops the procedure after 10 iterations.

Standard Errors

boottest does not calculate standard errors.

References


Examples

```r
## Not run:
library(fwildclusterboot)
data(voters)

lm_fit <- lm(proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
data = voters)

boot1 <- boottest(lm_fit,
B = 9999,
param = "treatment",
clustid = "group_id1"
)

boot2 <- boottest(lm_fit,
B = 9999,
param = "treatment",
clustid = c("group_id1", "group_id2")
)

boot3 <- boottest(lm_fit,
B = 9999,
param = "treatment",
clustid = c("group_id1", "group_id2"),
sign_level = 0.2,
seed = 8,
r = 2
)

# test treatment + ideology1 = 2

boot4 <- boottest(lm_fit,
B = 9999,
clustid = c("group_id1", "group_id2"),
param = c("treatment", "ideology1"),
R = c(1, 1),
r = 2
)

summary(boot1)
plot(boot1)

## End(Not run)
```

**boot_algo1**  
Fast wild cluster bootstrap algorithm
Description

function that implements the fast bootstrap algorithm as described in Roodman et al (2019)

Usage

boot_algo1(
  preprocessed_object,
  boot_iter,
  point_estimate,
  impose_null,
  r,
  sign_level,
  param,
  p_val_type,
  nthreads,
  type,
  full Enumeration,
  small_sample_correction,
  heteroskedastic,
  seed
)

Arguments

preprocessed_object  A list: output of the preprocess2 function.
boot_iter            number of bootstrap iterations
point_estimate       The point estimate of the test parameter from the regression model.
impose_null          If TRUE, the null is not imposed on the bootstrap distribution. This is what Roodman et al call the "WCU" bootstrap. With impose_null = FALSE, the null is imposed ("WCR").
r                   Shifts the null hypothesis.
sign_level          The significance level.
param               name of the test parameter.
p_val_type          type Type of p-value. By default "two-tailed". Other options: "equal-tailed", ">","<"
nthreads            The number of threads. Can be: a) an integer lower than, or equal to, the maximum number of threads; b) 0: meaning all available threads will be used; c) a number strictly between 0 and 1 which represents the fraction of all threads to use. The default is to use 50\ set permanently the number of threads used within this package using the function ...
type                character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default.
`boot_algo2`  

full_enumeration  
Is full enumeration employed? Full enum. is used if $N_G^2 < \text{boot_iter}$ for Mammen and Rademacher weights

small_sample_correction  
The small sample correction to be applied. See ssc().

heteroskedastic  
Logical - if TRUE, run a heteroskedastic. If FALSE, run wild cluster bootstrap.

seed  
Integer scalar. Either set via boottest()'s seed argument or inherited from R's global seed (set via set.seed)

Value  
A list of ...

---

`boot_algo2`  

*Fast wild cluster bootstrap algorithm*

**Description**  
function that implements the fast bootstrap algorithm as described in Roodman et al (2019)

**Usage**  

```r
boot_algo2(
  preprocessed_object,  
  boot_iter,  
  point_estimate,  
  impose_null,  
  r,  
  sign_level,  
  param,  
  p_val_type,  
  nthreads,  
  type,  
  full_enumeration,  
  small_sample_correction,  
  conf_int,  
  maxiter,  
  tol  
)
```

**Arguments**  

preprocessed_object  
A list: output of the preprocess2 function.

boot_iter  
number of bootstrap iterations
boot_ssc

point_estimate The point estimate of the test parameter from the regression model.

impose_null If TRUE, the null is not imposed on the bootstrap distribution. This is what Roodman et al call the "WCU" bootstrap. With impose_null = FALSE, the null is imposed ("WCR").

r Shifts the null hypothesis.

sign_level The significance level.

param name of the test parameter.

p_val_type type Type of p-value. By default "two-tailed". Other options: "equal-tailed", ">", "<"

nthreads The number of threads. Can be: a) an integer lower than, or equal to, the maximum number of threads; b) 0: meaning all available threads will be used; c) a number strictly between 0 and 1 which represents the fraction of all threads to use. The default is to use 50\ set permanently the number of threads used within this package using the function ...

type character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default.

full_enumeration Is full enumeration employed? Full enum. is used if N_G^2 < boot_iter for Mammen and Rademacher weights

small_sample_correction The small sample correction to be applied. See ssc().

conf_int Logical. Should confidence intervals be calculated (by test inversion)?

maxiter Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.

tol Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. 1e-6 by default.

Value

A list of ...

boot_ssc set the small sample correction factor applied in boottest()

Description

set the small sample correction factor applied in boottest()

Usage

boot_ssc(
  adj = TRUE,
  fixef.K = "none",
  cluster.adj = TRUE,
  cluster.df = "conventional"
)
cpp_get_nb_threads

Get maximum number of threads on hardware for open mp support

description

Get maximum number of threads on hardware for open mp support

usage

cpp_get_nb_threads()

create_data

Simulate Data

Description

Function simulates data for tests and examples with clustering variables and fixed-effects.

Usage

create_data(N, N_G1, icc1, N_G2, icc2, numb_fe1, numb_fe2, seed, weights)
Arguments

- **N**  
  number of observations

- **N_G1**  
  A scalar. number of clusters for clustering variable 1

- **icc1**  
  A scalar between 0 and 1. intra-cluster correlation for clustering variable 1

- **N_G2**  
  A scalar. number of clusters for clustering variable 2

- **icc2**  
  A scalar between 0 and 1. intra-cluster correlation for clustering variable 2

- **numb_fe1**  
  A scalar. Number of fixed effect for first factor variable

- **numb_fe2**  
  A scalar. Number of fixed effect for second factor variable

- **seed**  
  An integer. Set the random seed

- **weights**  
  Possible regression weights to be used in estimation

Value

A simulated *data.frame* with specified numbers of clusters, intra-cluster correlations and dimensionality of fixed effects.

---

eigenMapMatMult  
*Matrix Multiplication via Eigen*

Description

Matrix Multiplication via Eigen

Usage

eigenMapMatMult(A, B, nthreads)

Arguments

- **A**  
  A matrix.

- **B**  
  A matrix.

- **nthreads**  
  Integer. Number of threads to use for matrix multiplication.

Value

A matrix
get_bootstrapped_pvalue  

**get bootstrapped p-value based on bootstrapped t-stats**

**Description**

get bootstrapped p-value based on bootstrapped t-stats

**Usage**

get_bootstrapped_pvalue(p_val_type, t_stat, t_boot)

**Arguments**

- **p_val_type**  Character vector of length 1. Type of p-value. Options include "two-tailed", "equal-tailed", ">" and "<".
- **t_stat**  The original t-statistic
- **t_boot**  The bootstrapped t-statistics

**Value**

A bootstrapped p-value

---

get_seed  

**creates an integer based on the global random seed set via set.seed() for using set.seed() for controlling rcpp’s seed, see this blog post http://rorynolan.rbind.io/2018/09/30/rcsetseed/**

**Description**

creates an integer based on the global random seed set via set.seed() for using set.seed() for controlling rcpp’s seed, see this blog post http://rorynolan.rbind.io/2018/09/30/rcsetseed/

**Usage**

get_seed()
get_ssc

*Compute small sample adjustment factors*

**Description**

Compute small sample adjustment factors

**Usage**

```r
get_ssc(boot_ssc_object, N, k, G, vcov_sign, heteroskedastic = FALSE)
```

**Arguments**

- `boot_ssc_object`: An object of type 'boot_ssc.type'
- `N`: The number of observations
- `k`: The number of estimated parameters
- `G`: The number of clusters
- `vcov_sign`: A vector that helps create the covariance matrix
- `heteroskedastic`: Heteroskedastic wild bootstrap? FALSE by default. If TRUE, cluster adjustments via G and vcov_sign will be ignored

**Value**

A small sample adjustment factor

---

glance.boottest

*S3 method to glance at objects of class boottest*

**Description**

S3 method to glance at objects of class boottest

**Usage**

```r
## S3 method for class 'boottest'
glance(x, ...)
```

**Arguments**

- `x`: object of type boottest
- `...`: Further arguments passed to or from other methods.
mbootest

Value

A single row summary "glance" of an object of type mbootest - lists characteristics of the input regression model

mbootest  Arbitrary Linear Hypothesis Testing for Regression Models via Wald-Tests

Description

mbootest is a S3 method that allows for arbitrary linear hypothesis testing for objects of class lm, fixest, felm

Usage

mbootest(object, ...)

Arguments

object   An object of type lm, fixest or felm

other arguments

Value

An object of class mbootest.

Setting Seeds

To guarantee reproducibility, you can either use bootest()’s seed function argument, or set a global random seed via

- set.seed() when using
  1. the lean algorithm (via boot_algo = "R-lean"), 2) the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via boot_algo = "R" with Mammen weights or 4) boot_algo = "WildBootTests.jl"
- dqrng::dqset.seed() when using boot_algo = "R" for Rademacher, Webb or Normal weights

References


mboottest.felm


See Also

mboottest.lm mboottest.felm mboottest.fixest

mboottest.felm

Fast wild cluster bootstrap inference for joint hypotheses for object of class felm

Description

mboottest.felm is a S3 method that allows for fast wild cluster bootstrap inference of multivariate hypotheses for objects of class felm by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019.

Usage

## S3 method for class 'felm'
mboottest(
  object,
  clustid,
  B,
  R,
  r = rep(0, nrow(R)),
  bootcluster = "max",
  fe = NULL,
  seed = NULL,
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  floattype = "Float64",
  getauxweights = FALSE,
  teststat_boot = FALSE,
  maxmatsize = NULL,
  bootstrapc = FALSE,
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df = "conventional"),
  ...
)

)
Arguments

object
An object of class felm

clustid
A character vector or rhs formula containing the names of the cluster variables

B
Integer. The number of bootstrap iterations. When the number of clusters is low, increasing B adds little additional runtime.

R
Hypothesis Vector or Matrix giving linear combinations of coefficients. Must be either a vector of length k or a matrix of dimension q x k, where q is the number of joint hypotheses and k the number of estimated coefficients.

r
A vector of length q, where q is the number of tested hypotheses. Shifts the null hypothesis H0: param = r vs H1: param ! = r. If not provided, a vector of zeros of length q.

bootcluster
A character vector or rhs formula of length 1. Specifies the bootstrap clustering variable or variables. If more than one variable is specified, then bootstrapping is clustered by the intersections of clustering implied by the listed variables. To mimic the behavior of stata’s boottest command, the default is to cluster by the intersection of all the variables specified via the clustid argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.

fe
A character vector or rhs formula of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, fe needs to be NULL.

seed
An integer. Allows to set a random seed. For details, see below.

type
character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher and Mammen distribution, if the number of replications B exceeds the number of possible draw combinations, $2^\text{(#number of clusters)}$, then boottest() will use each possible combination once (enumeration).

impose_null
Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)

p_val_type
Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".

tol
Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. Relative tolerance of 1e-6 by default.

floattype
Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit?

getauxweights
Logical. FALSE by default. Whether to save auxiliary weight matrix (v)

teststat_boot
Logical. Should bootstrapped test statistics be returned?
maxmatsize: NULL by default = no limit. Else numeric scalar to set the maximum size of auxiliary weight matrix (v), in gigabytes.

bootstrapc: Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t.

ssc: An object of class boot_ssc.type obtained with the function boot_ssc. Represents how the small sample adjustments are computed. The defaults are adj = TRUE, fixef.K = “none”, ssc = NULL. You can find more details in the help file for boot_ssc(). The function is purposefully designed to mimic fixest’s ssc function.

Value

An object of class mboottest

p_val: The bootstrap p-value.

N: Sample size. Might differ from the regression sample size if the cluster variables contain NA values.

boot_iter: Number of Bootstrap Iterations.

clustid: Names of the cluster Variables.

N_G: Dimension of the cluster variables as used in boottest.

sign_level: Significance level used in boottest.

type: Distribution of the bootstrap weights.

impose_null: Whether the null was imposed on the bootstrap dgp or not.

R: The vector "R" in the null hypothesis of interest Rbeta = r.

r: The scalar "r" in the null hypothesis of interest Rbeta = r.

point_estimate: R'beta. A scalar: the constraints vector times the regression coefficients.

teststat_stat: The ‘original’ regression test statistics.

teststat_boot: All bootstrap t-statistics.

regression: The regression object used in boottest.

call: Function call of boottest.

boot_algo: The employed bootstrap algorithm.

nthreads: The number of threads employed.

internal_seed: The integer value -inherited from set.seed() - used within boottest() to set the random seed in either R or Julia. If NULL, no internal seed was created.

Setting Seeds

To guarantee reproducibility, you can either use boottest()’s seed function argument, or set a global random seed via

• set.seed() when using
  1. the lean algorithm (via boot_algo = "R-lean") including the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via boot_algo = "R" with Mammen weights or
  3. boot_algo = "WildBootTests.jl"
• dqrng::dqset.seed() when using boot_algo = "R" for Rademacher, Webb or Normal weights
mboottest.fixest

References


Examples

```r
## Not run:
library(lfe)
library(clubSandwich)
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
)
generics::tidy(wboottest)
## End(Not run)
```

mboottest.fixest

Fast wild cluster bootstrap inference for joint hypotheses for object of class fixest

Description

mboottest.fixest is a S3 method that allows for fast wild cluster bootstrap inference of multivariate hypotheses for objects of class fixest by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019.

Usage

```r
## S3 method for class 'fixest'
mboottest(
    object,
    clustid,
)```
B,
R,
\( r = \text{rep}(0, \text{nrow}(R)) \),
\text{bootcluster} = "\text{max}",
\text{fe} = NULL,
\text{seed} = NULL,
\text{type} = "\text{rademacher}",
\text{impose_null} = \text{TRUE},
\text{p_val_type} = "\text{two-tailed}",
\text{tol} = 1e-06,
\text{floattype} = "\text{Float64}",
\text{getauxweights} = \text{FALSE},
\text{teststat_boot} = \text{FALSE},
\text{maxmatsize} = \text{NULL},
\text{bootstrapc} = \text{FALSE},
\text{ssc} = \text{boot_ssc(adj = \text{TRUE}, fixef.K = "\text{none}", cluster.adj = \text{TRUE}, cluster.df = "\text{conventional}")},
 ...
)

\textbf{Arguments}

\begin{itemize}
  \item \textbf{object} \hspace{1cm} \text{An object of class feols}
  \item \textbf{clustid} \hspace{1cm} \text{A character vector or rhs formula containing the names of the cluster variables}
  \item \textbf{B} \hspace{1cm} \text{Integer. The number of bootstrap iterations. When the number of clusters is low, increasing B adds little additional runtime.}
  \item \textbf{R} \hspace{1cm} \text{Hypothesis Vector or Matrix giving linear combinations of coefficients. Must be either a vector of length k or a matrix of dimension q x k, where q is the number of joint hypotheses and k the number of estimated coefficients.}
  \item \textbf{r} \hspace{1cm} \text{A vector of length q, where q is the number of tested hypotheses. Shifts the null hypothesis H0: param = r vs H1: param != r. If not provided, a vector of zeros of length q.}
  \item \textbf{bootcluster} \hspace{1cm} \text{A character vector or rhs formula of length 1. Specifies the bootstrap clustering variable or variables. If more than one variable is specified, then bootstrapping is clustered by the intersections of clustering implied by the listed variables. To mimic the behavior of stata's boottest command, the default is to cluster by the intersection of all the variables specified via the clustid argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.}
  \item \textbf{fe} \hspace{1cm} \text{A character vector or rhs formula of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, fe needs to be NULL.}
  \item \textbf{seed} \hspace{1cm} \text{An integer. Allows to set a random seed. For details, see below.}
\end{itemize}
type character or function. The character string specifies the type of bootstrap to use:
One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively,
type can be a function(n) for drawing wild bootstrap factors. "rademacher" by
deault. For the Rademacher and Mammen distribution, if the number of repli-
cations B exceeds the number of possible draw ombinations, \(2^{(#\text{number of}}
clusters)}\), then boottest() will use each possible combination once (enumera-
tion).

impose_null Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not.
Null imposed \((\text{WCR})\) by default. If FALSE, the null is not imposed \((\text{WCU})\)

p_val_type Character vector of length 1. Type of p-value. By default "two-tailed". Other
options include "equal-tailed", ">" and "<".

tol Numeric vector of length 1. The desired accuracy (convergence tolerance) used
in the root finding procedure to find the confidence interval. Relative tolerance
of 1e-6 by default.

floattype Float64 by default. Other option: Float32. Should floating point numbers in
Julia be represented as 32 or 64 bit?

getauxweights Logical. FALSE by default. Whether to save auxilliary weight matrix \((v)\)

teststat_boot Logical. Should bootstrapped test statistics be returned?

maxmatsize NULL by default = no limit. Else numeric scalar to set the maximum size of
auxilliary weight matrix \((v)\), in gigabytes

bootstrapc Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of
bootstrap-t

ssc An object of class boot_ssc.type obtained with the function boot_ssc. Repre-
sents how the small sample adjustments are computed. The defaults are \(\text{adj} = \text{TRUE}, \text{fixef.K} = \text{"none"},\)
You can find more details in the help file for boot_ssc(). The function is pur-
posefully designed to mimic fixest’s ssc function.

Value
An object of class mboottest

p_val The bootstrap p-value.
N Sample size. Might differ from the regression sample size if the cluster variables
contain NA values.
boot_iter Number of Bootstrap Iterations.
clustid Names of the cluster Variables.
N_G Dimension of the cluster variables as used in boottest.
sign_level Significance level used in boottest.
type Distribution of the bootstrap weights.
impose_null Whether the null was imposed on the bootstrap dgp or not.
R The vector "R" in the null hypothesis of interest Rbeta = r.
r The scalar "r" in the null hypothesis of interest Rbeta = r.
point_estimate R’beta. A scalar: the constraints vector times the regression coefficients.
teststat_stat The ‘original’ regression test statistics.
teststat_boot All bootstrap t-statistics.
regression The regression object used in boottest.
call Function call of boottest.
boot_algo The employed bootstrap algorithm.
nthreads The number of threads employed.
internal_seed The integer value -inherited from set.seed() - used within boottest() to set the random seed in either R or Julia. If NULL, no internal seed was created.

Setting Seeds

To guarantee reproducibility, you can either use boottest()’s seed function argument, or set a global random seed via

- set.seed() when using
  1. the lean algorithm (via boot_algo = "R-lean") including the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via boot_algo = "R" with Mammen weights or
  3. boot_algo = "WildBootTests.jl"
- dqrng::dqset.seed() when using boot_algo = "R" for Rademacher, Webb or Normal weights

References


Examples

```r
## Not run:
library(fwildclusterboot)
library(clubSandwich)
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
```
mboottest.lm

### Description

*mboottest.lm* is a S3 method that allows for fast wild cluster bootstrap inference of multivariate hypotheses for objects of class `lm` by implementing the fast wild bootstrap algorithm developed in Roodman et al., 2019.

### Usage

```r
## S3 method for class 'lm'
mboottest(
  object,  
clustid,  
B,  
R,  
r = rep(0, nrow(R)),  
bootcluster = "max",  
seed = NULL,  
type = "rademacher",  
impose_null = TRUE,  
p_val_type = "two-tailed",  
tol = 1e-06,  
floattype = "Float64",  
getauxweights = FALSE,  
teststat_boot = FALSE,  
maxmatsize = NULL,  
bootstrapc = FALSE,  
ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df = "conventional"),
  ...
)
```

### Arguments

- **object**: An object of class `lm`
- **clustid**: A character vector or rhs formula containing the names of the cluster variables
- **B**: Integer. The number of bootstrap iterations. When the number of clusters is low, increasing B adds little additional runtime.
R hypothesis Vector or Matrix giving linear combinations of coefficients. Must be either a vector of length k or a matrix of dimension q x k, where q is the number of joint hypotheses and k the number of estimated coefficients.

r A vector of length q, where q is the number of tested hypotheses. Shifts the null hypothesis H0: param = r vs H1: param != r. If not provided, a vector of zeros of length q.

bootcluster A character vector or rhs formula of length 1. Specifies the bootstrap clustering variable or variables. If more than one variable is specified, then bootstrapping is clustered by the intersections of clustering implied by the listed variables. To mimic the behavior of stata’s boottest command, the default is to cluster by the intersection of all the variables specified via the clustid argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.

seed An integer. Allows to set a random seed. For details, see below.

type character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher and Mammen distribution, if the number of replications B exceeds the number of possible combinations, 2^(#number of clusters), then boottest() will use each possible combination once (enumeration).

impose_null Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)

p_val_type Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".

tol Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. Relative tolerance of 1e-6 by default.

floattype Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit?

getauxweights Logical. FALSE by default. Whether to save auxilliary weight matrix (v)

teststat_boot Logical. Should bootstrapped test statistics be returned?

maxmatsize NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes

bootstrapc Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t

ssc An object of class boot_ssc.type obtained with the function boot_ssc. Represents how the small sample adjustments are computed. The defaults are adj = TRUE, fixef.K = "none", You can find more details in the help file for boot_ssc(). The function is purposefully designed to mimic fixest’s ssc function.

... Further arguments passed to or from other methods.
Value

An object of class `mboottest`

- **p_val**: The bootstrap p-value.
- **N**: Sample size. Might differ from the regression sample size if the cluster variables contain NA values.
- **boot_iter**: Number of Bootstrap Iterations.
- **clustid**: Names of the cluster Variables.
- **N_G**: Dimension of the cluster variables as used in boottest.
- **sign_level**: Significance level used in boottest.
- **type**: Distribution of the bootstrap weights.
- **impose_null**: Whether the null was imposed on the bootstrap dgp or not.
- **R**: The vector "R" in the null hypothesis of interest Rbeta = r.
- **r**: The scalar "r" in the null hypothesis of interest Rbeta = r.
- **point_estimate**: R'beta. A scalar: the constraints vector times the regression coefficients.
- **teststat_stat**: The 'original' regression test statistics.
- **teststat_boot**: All bootstrap t-statistics.
- **regression**: The regression object used in boottest.
- **call**: Function call of boottest.
- **boot_algo**: The employed bootstrap algorithm.
- **nthreads**: The number of threads employed.
- **internal_seed**: The integer value -inherited from set.seed() - used within boottest() to set the random seed in either R or Julia. If NULL, no internal seed was created.

Setting Seeds

To guarantee reproducibility, you can either use boottest()’s seed function argument, or set a global random seed via

- **set.seed()** when using
  1. the lean algorithm (via boot_algo = "R-lean") including the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via boot_algo = "R" with Mammen weights or
  3. boot_algo = "WildBootTests.jl"
- **dqrng::dqset.seed()** when using boot_algo = "R" for Rademacher, Webb or Normal weights

References


Examples

```r
## Not run:
library(clubSandwich)
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R,
  )
generics::tidy(wboottest)
## End(Not run)
```

---

**model_matrix**

enhanced model.matrix functionalities

**Description**

enhanced model.matrix functionalities

**Usage**

```r
model_matrix(object, ...)
```

**Arguments**

- `object` An object of class `lm` or `felm`
- `...` Other arguments
### Description

Enhanced model.matrix for objects of type `felm`

### Usage

```r
## S3 method for class 'felm'
model_matrix(object, type, collin.rm = TRUE, ...)
```

### Arguments

- `object`: An object of class `felm`
- `type`: 'rhs' for right-hand side variables, 'fixef' for fixed effects
- `collin.rm`: Should collinear variables be dropped?
- `...`: Other arguments

---

### Description

Enhanced model.matrix for objects of type `fixest`

### Usage

```r
## S3 method for class 'fixest'
model_matrix(object, type, collin.rm = TRUE, ...)
```

### Arguments

- `object`: An object of class `fixest`
- `type`: rhs, lhs or fixef
- `collin.rm`: Should collinear variables be dropped?
- `...`: Other arguments
**model_matrix.lm**  
*Enhanced model.matrix for objects of type lm*

**Description**

Enhanced model.matrix for objects of type lm

**Usage**

```r
## S3 method for class 'lm'
model_matrix(object, collin.rm = TRUE, ...)
```

**Arguments**

- `object`: An object of class lm
- `collin.rm`: Should collinear variables be dropped?
- `...`: Other arguments

---

**plot.boottest**  
*Plot the bootstrap distribution of t-statistics*

**Description**

Plot the bootstrap distribution of t-statistics

**Usage**

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

**Arguments**

- `x`: An object of type boottest
- `...`: Further arguments passed to or from other methods.

**Value**

A plot of bootstrap t-statistics under different null hypotheses
**Description**

`preprocess2` is an S3 method that fetches data from several model objects for use with `boottest()`.

**Usage**

`preprocess2(object, ...)`

**Arguments**

- `object` An object of type `lm`, `fixest`, `felm` or `ivreg`
- `...` other arguments

**Value**

An object of class `preprocess2`.

---

**setBoottest_boot_algo** Sets the bootstrap algo to be run via `boottest()` and `waldboottest()`

**Description**

Sets the bootstrap algo to be run via `boottest()` and `waldboottest()`

**Usage**

`setBoottest_boot_algo(boot_algo)`

**Arguments**

- `boot_algo` Character scalar. Either 'R' or 'WildBootTests.jl'. Default is 'R'

**Value**

No return value
summary.boottest

S3 method to summarize objects of class boottest

Description

S3 method to summarize objects of class boottest

Usage

## S3 method for class 'boottest'
summary(object, digits = 3, ...)

Arguments

object object of type boottest
digits rounding of output. 3 by default
... Further arguments passed to or from other methods.

Value

Returns result summaries for objects of type boottest

summary.mboottest

S3 method to summarize objects of class mboottest

Description

S3 method to summarize objects of class mboottest

Usage

## S3 method for class 'mboottest'
summary(object, digits = 3, ...)

Arguments

object object of type mboottest
digits rounding of output. 3 by default
... Further arguments passed to or from other methods.

Value

Returns result summaries for objects of type mboottest
tidy.boottest

S3 method to summarize objects of class boottest into tidy data.frame

Description

S3 method to summarize objects of class boottest into tidy data.frame

Usage

## S3 method for class 'boottest'
tidy(object, ...)

Arguments

object object of type boottest
...
Further arguments passed to or from other methods.

Value

A tidy data.frame with estimation results for objects of type boottest

tidy.mboottest

S3 method to summarize objects of class mboottest into tidy data.frame

Description

S3 method to summarize objects of class mboottest into tidy data.frame

Usage

## S3 method for class 'mboottest'
tidy(object, ...)

Arguments

object object of type mboottest
...
Further arguments passed to or from other methods.

Value

A tidy data.frame with estimation results for objects of type mboottest
**Description**

Transform vectors of all types safely to integer vectors

**Usage**

to_integer(vec)

**Arguments**

vec  
A vector

**Value**

An integer vector

---

**voters**

*Random example data set*

**Description**

Random example data set

**Usage**

data(voters)

**Format**

An object of class `data.frame` with 300 rows and 13 columns.

**Examples**

data(voters)
Index

* datasets
  voters, 48
  .onLoad, 3

boot_algo1, 23
boot_algo2, 25
boot_ssc, 6, 11, 17, 21, 26, 34, 37, 40
boottest, 3
boottest.felm, 4, 4
boottest.fixest, 4, 9
boottest.ivreg, 4, 15
boottest.lm, 4, 19

cpp_get_nb_threads, 27
create_data, 27

eigenMapMatMult, 28

get_bootstrap_pvalue, 29
get_seed, 29
get_ssc, 30
glance.boottest, 30

mboottest, 31
mboottest.felm, 32, 32
mboottest.fixest, 32, 35
mboottest.lm, 32, 39
model_matrix, 42
model_matrix.felm, 43
model_matrix.fixest, 43
model_matrix.lm, 44

plot.boottest, 44
preprocess2, 45

setBoottest_boot_algo, 45
ssc, 6, 11, 17, 21, 34, 37, 40
summary.boottest, 46
summary.mboottest, 46

tidy.boottest, 47

tidy.mboottest, 47
to_integer, 48

voters, 48