Package ‘fwildclusterboot’

February 26, 2023

Title Fast Wild Cluster Bootstrap Inference for Linear Models

Version 0.13.0

Description Implementation of fast algorithms for wild cluster bootstrap inference developed in 'Roodman et al' (2019, 'STATA' Journal, <doi:10.1177/1536867X19830877>) and 'MacKinnon et al' (2022), which makes it feasible to quickly calculate bootstrap test statistics based on a large number of bootstrap draws even for large samples. Multiple bootstrap types as described in 'MacKinnon, Nielsen & Webb' (2022) are supported.
Further, '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.
Last, 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, dreamerr, Formula, generics, dqrng, gtools, Matrix, JuliaConnectoR, MASS, Rcpp, summclust, rlang

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

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

LinkingTo Rcpp,RcppArmadillo, RcppEigen

VignetteBuilder knitr

Language en-US

SystemRequirements Version Requirements to run the wild bootstrap through Julia - Julia (>= 1.8), WildBootTests.jl (>=0.9). Julia is downloadable via the official Julia website (https://julialang.org/downloads/), WildBootTests.jl via Julia's package manager (https://docs.julialang.org/en/v1/stdlib/Pkg/) or its github repository (https://github.com/droodman/WildBootTests.jl)

Config/testthat/edition 3

NeedsCompilation yes

Author Alexander Fischer [aut, cre],
David Roodman [aut],
Achim Zeileis [ctb] (Author of included sandwich fragments),
Nathaniel Graham [ctb] (Contributor to included sandwich fragments),
Susanne Koell [ctb] (Contributor to included sandwich fragments),
Laurent Berge [ctb] (Author of included fixest fragments),
Sebastian Krantz [ctb]

Maintainer Alexander Fischer <alexander-fischer1801@t-online.de>

Repository CRAN

Date/Publication 2023-02-26 01:00:13 UTC

R topics documented:

`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

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

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

Stata, Julia and Python Implementations

The fast wild cluster bootstrap algorithms are further implemented in the following software packages:

- Stata:boottest
- Julia:WildBootTests.jl
- Python:wildboottest

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (https://ideas.repec.org/p/qed/wpaper/1406.html)

MacKinnon, James G., Morten Ørregaard Nielsen, and Matthew D. Webb. Fast and reliable jackknife and bootstrap methods for cluster-robust inference. No. 1485. 2022.

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. "Reworking wild bootstrap based inference for clustered errors" . No. 1315. Queen's Economics Department Working Paper, 2013.

See Also

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

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
  proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
summary(boot)
```

```
print(boot)
plot(boot)
nobs(boot)
pval(boot)
confint(boot)
generics::tidy(boot)
```

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 fast wild bootstrap algorithms as developed in Roodman et al., 2019 and MacKinnon, Nielsen & Webb (2022).

Usage

```
## S3 method for class 'felm'
boottest(
  object,
  param,
  B,
  clustid = NULL,
  bootcluster = "max",
  fe = NULL,
  conf_int = TRUE,
  R = NULL,
  r = 0,
  beta0 = NULL,
  sign_level = 0.05,
  type = "rademacher",
  impose_null = TRUE,
  bootstrap_type = "fnw11",
  p_val_type = "two-tailed",
  tol = 1e-06,
  maxiter = 10,
  sampling = "dqrng",
  nthreads = getBoottest_nthreads(),
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
    "conventional"),
  engine = getBoottest_engine(),
  floattype = "Float64",
  maxmatsize = FALSE,
  bootstrapc = 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.

`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.

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

`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 boostrap 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)`

`bootstrap_type` Determines which wild cluster bootstrap type should be run. Options are "fnw11","11", "13", "31" and "33" for the wild cluster bootstrap and "11" and "31" for the heteroskedastic bootstrap. For more information, see the details section. "fnw11" is the default for the cluster bootstrap, which runs a "11" type wild cluster bootstrap via the algorithm outlined in "fast and wild" (Roodman et al (2019)). "11" is the default for the heteroskedastic bootstrap.

`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.

`sampling` 'dqrng' or 'standard'. If 'dqrng', the 'dqrng' package is used for random number generation (when available). If 'standard', functions from the 'stats' package are used when available. This argument is mostly a convenience to control random number generation in a wrapper package around `fwildclusterboot`, `wildrwolf`. I recommend to use the fast' option.

`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", cluster.adj = "TRUE", cluster.df = "conventional". You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest's `fixest::ssc()` function.

`engine` 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_engine()` function.

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

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

`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.

`t_boot` All bootstrap t-statistics.

`regression` The regression object used in boottest.

`call` Function call of boottest.

`engine` The employed bootstrap algorithm.

`nthreads` The number of threads employed.

Setting Seeds

To guarantee reproducibility, you need to set a global random seed via

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

Stata, Julia and Python Implementations

The fast wild cluster bootstrap algorithms are further implemented in the following software packages:

- Stata:boottest
- Julia:WildBootTests.jl
- Python:wildboottest

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (`https://ideas.repec.org/p/qed/wpaper/1406.html`)

MacKinnon, James G., Morten Ørregaard Nielsen, and Matthew D. Webb. Fast and reliable jackknife and bootstrap methods for cluster-robust inference. No. 1485. 2022.

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters. " The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. Reworking wild bootstrap based inference for clustered errors. No. 1315. Queen's Economics Department Working Paper, 2013.

Examples

```
## Not run:
  requireNamespace("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,
    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)
  print(boot1)
  plot(boot1)
  nobs(boot1)
  pval(boot1)
  confint(boot1)
  generics::tidy(boot1)

# run different bootstrap types following MacKinnon, Nielsen & Webb (2022):

# default: the fnw algorithm
boot_fnw11 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "fnw11"
)

# WCR 31
boot_WCR31 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "31"
)

# WCU33
boot_WCR31 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "33",
  impose_null = FALSE
)

## End(Not run)
```

boottest.fixest *Fast wild cluster bootstrap inference for object of class fixest*

Description

boottest.fixest is a S3 method that allows for fast wild cluster bootstrap inference for objects of class fixest by implementing fast wild bootstrap algorithms as developed in Roodman et al., 2019 and MacKinnon, Nielsen & Webb (2022).

Usage

```
## S3 method for class 'fixest'
boottest(
  object,
  param,
  B,
  clustid = NULL,
  bootcluster = "max",
  fe = NULL,
  sign_level = 0.05,
  conf_int = TRUE,
  R = NULL,
  r = 0,
  beta0 = NULL,
  type = "rademacher",
  impose_null = TRUE,
  bootstrap_type = "fnw11",
  p_val_type = "two-tailed",
  tol = 1e-06,
  maxiter = 10,
  sampling = "dqrng",
  nthreads = getBoottest_nthreads(),
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
    "conventional"),
  engine = getBoottest_engine(),
  floattype = "Float64",
  maxmatsize = FALSE,
  bootstrapc = 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.

`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.

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

`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 boostrap 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)`

`bootstrap_type` Determines which wild cluster bootstrap type should be run. Options are "fnw11","11", "13", "31" and "33" for the wild cluster bootstrap and "11" and "31" for the heteroskedastic bootstrap. For more information, see the details section. "fnw11" is the default for the cluster bootstrap, which runs a "11" type wild cluster bootstrap via the algorithm outlined in "fast and wild" (Roodman et al (2019)). "11" is the default for the heteroskedastic bootstrap.

`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.

`sampling` 'dqrng' or 'standard'. If 'dqrng', the 'dqrng' package is used for random number generation (when available). If 'standard', functions from the 'stats' package are used when available. This argument is mostly a convenience to control random number generation in a wrapper package around `fwildclusterboot`, `wildrwolf`. I recommend to use the fast' option.

`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", cluster.adj = "TRUE", cluster.df = "conventional"`. You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest's `fixest::ssc()` function.

`engine` Character scalar. Either "R", "R-lean" or "WildBootTests.jl". Controls if `boottest()` should run via its native R implementation or `WildBootTests.jl`. "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. The "R-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. You can set the employed algorithm globally by using the `setBoottest_engine()` function.

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

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

`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.

`t_boot` All bootstrap t-statistics.

`regression` The regression object used in boottest.

`call` Function call of boottest.

`engine` The employed bootstrap algorithm.

`nthreads` The number of threads employed.

Setting Seeds

To guarantee reproducibility, you need to set a global random seed via

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

Stata, Julia and Python Implementations

The fast wild cluster bootstrap algorithms are further implemented in the following software packages:

- Stata:boottest
- Julia:WildBootTests.jl
- Python:wildboottest

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (`https://ideas.repec.org/p/qed/wpaper/1406.html`)

MacKinnon, James G., Morten Ørregaard Nielsen, and Matthew D. Webb. Fast and reliable jackknife and bootstrap methods for cluster-robust inference. No. 1485. 2022.

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters. " The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. Reworking wild bootstrap based inference for clustered errors. No. 1315. Queen's Economics Department Working Paper, 2013.

Examples

```
## Not run:
requireNamespace("fixest")
requireNamespace("fwildclusterboot")
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,
  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)
print(boot1)
plot(boot1)
nobs(boot1)
pval(boot1)
confint(boot1)
generics::tidy(boot1)

# run different bootstrap types following MacKinnon, Nielsen & Webb (2022):

# default: the fnw algorithm
boot_fnw11 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "fnw11"
)

# WCR 31
boot_WCR31 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "31"
)

# WCU33
boot_WCR31 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "33",
  impose_null = FALSE
)


## 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

```
## S3 method for class 'ivreg'
boottest(
  object,
  clustid,
  param,
  B,
  bootcluster = "max",
  conf_int = TRUE,
  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,
  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` Integer. The number of bootstrap iterations. When the number of clusters is low, increasing B adds little additional runtime

`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.

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

`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

`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 boostrap 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 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. 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)

`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

`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", cluster.adj = "TRUE", cluster.df = "conventional"`. You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest's `fixest::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 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.

`call` Function call of boottest.

`engine` The employed bootstrap algorithm.

`nthreads` The number of threads employed.

Setting Seeds

To guarantee reproducibility, you need to set a global random seed via `set.seed()`

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (`https://ideas.repec.org/p/qed/wpaper/1406.html`)

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. Reworking wild bootstrap based inference for clustered errors. No. 1315. Queen's Economics Department Working Paper, 2013.

Examples

```
## Not run:
requireNamespace("ivreg")
requireNamespace("fwildclusterboot")

# drop all NA values from SchoolingReturns
SchoolingReturns <- na.omit(SchoolingReturns)
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)
print(boot_ivreg)
plot(boot_ivreg)
nobs(boot_ivreg)
pval(boot_ivreg)
confint(boot_ivreg)
generics::tidy(boot_ivreg)

## End(Not run)
```

`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 fast wild bootstrap algorithms as developed in Roodman et al., 2019 and MacKinnon, Nielsen & Webb (2022).

Usage

```
## S3 method for class 'lm'
boottest(
  object,
  param,
  B,
  clustid = NULL,
  bootcluster = "max",
  conf_int = TRUE,
  R = NULL,
  r = 0,
  beta0 = NULL,
  sign_level = 0.05,
  type = "rademacher",
  impose_null = TRUE,
  bootstrap_type = "fnw11",
  p_val_type = "two-tailed",
  tol = 1e-06,
  maxiter = 10,
  sampling = "dqrng",
  nthreads = getBoottest_nthreads(),
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
    "conventional"),
  engine = getBoottest_engine(),
  floattype = "Float64",
  maxmatsize = FALSE,
  bootstrapc = 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, 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.

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

`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 boostrap 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)`

`bootstrap_type` Determines which wild cluster bootstrap type should be run. Options are "fnw11","11", "13", "31" and "33" for the wild cluster bootstrap and "11" and "31" for the heteroskedastic bootstrap. For more information, see the details section. "fnw11" is the default for the cluster bootstrap, which runs a "11" type wild cluster bootstrap via the algorithm outlined in "fast and wild" (Roodman et al (2019)). "11" is the default for the heteroskedastic bootstrap.

`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.

`sampling` 'dqrng' or 'standard'. If 'dqrng', the 'dqrng' package is used for random number generation (when available). If 'standard', functions from the 'stats' package are used when available. This argument is mostly a convenience to control random number generation in a wrapper package around `fwildclusterboot`, `wildrwolf`. I recommend to use the fast' option.

`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", cluster.adj = "TRUE", cluster.df = "conventional"`. You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest's `fixest::ssc()` function.

`engine` Character scalar. Either "R", "R-lean" or "WildBootTests.jl". Controls if `boottest()` should run via its native R implementation or `WildBootTests.jl`. "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. The "R-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. You can set the employed algorithm globally by using the `setBoottest_engine()` function.

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

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

`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.
`t_boot`	All bootstrap t-statistics.
`regression`	The regression object used in boottest.
`call`	Function call of boottest.
`engine`	The employed bootstrap algorithm.
`nthreads`	The number of threads employed.

Setting Seeds

To guarantee reproducibility, you need to set a global random seed via

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

Via the `engine` function argument, it is possible to specify different variants of the wild cluster bootstrap, and if the algorithm should be run via R or `WildBootTests.jl`.

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.

Stata, Julia and Python Implementations

The fast wild cluster bootstrap algorithms are further implemented in the following software packages:

- Stata:boottest
- Julia:WildBootTests.jl
- Python:wildboottest

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (https://ideas.repec.org/p/qed/wpaper/1406.html)

MacKinnon, James G., Morten Ørregaard Nielsen, and Matthew D. Webb. Fast and reliable jackknife and bootstrap methods for cluster-robust inference. No. 1485. 2022.

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. Reworking wild bootstrap based inference for clustered errors. No. 1315. Queen's Economics Department Working Paper, 2013.

Examples

```
## Not run:
requireNamespace("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,
  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)
print(boot1)
plot(boot1)
nobs(boot1)
pval(boot1)
confint(boot1)
generics::tidy(boot1)

# run different bootstrap types following MacKinnon, Nielsen & Webb (2022):

# default: the fnw algorithm
boot_fnw11 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "fnw11"
)

# WCR 31
boot_WCR31 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "31"
)

# WCU33
boot_WCR31 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1",
  bootstrap_type = "33",
  impose_null = FALSE
)


## End(Not run)
```

boot_aggregate *Simple tool that aggregates the value of CATT coefficients in staggered difference-in-difference setups with inference based on a wild cluster bootstrap (see details) - similar to* `fixest::aggregate()`

Description

This is a function helping to replicate the estimator from Sun and Abraham (2021, Journal of Econometrics). You first need to perform an estimation with cohort and relative periods dummies (typically using the function i), this leads to estimators of the cohort average treatment effect on the treated (CATT). Then you can use this function to retrieve the average treatment effect on each relative period,or for any other way you wish to aggregate the CATT.

Usage

```
boot_aggregate(
  x,
  agg,
  full = FALSE,
  use_weights = TRUE,
  clustid = NULL,
  B,
  bootcluster = "max",
  fe = NULL,
  sign_level = 0.05,
  beta0 = NULL,
  type = "rademacher",
  impose_null = TRUE,
  bootstrap_type = "fnw11",
  p_val_type = "two-tailed",
  nthreads = getBoottest_nthreads(),
  tol = 1e-06,
  maxiter = 10,
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
    "conventional"),
  engine = getBoottest_engine(),
  floattype = "Float64",
  maxmatsize = FALSE,
  bootstrapc = FALSE,
  getauxweights = FALSE,
  sampling = "dqrng",
  ...
)
```

Arguments

x An object of type fixest estimated using `sunab()`

`agg` A character scalar describing the variable names to be aggregated, it is pattern-based. All variables that match the pattern will be aggregated. It must be of the form `"(root)"`, the parentheses must be there and the resulting variable name will be `"root"`. You can add another root with parentheses: `"(root1)regex(root2)"`, in which case the resulting name is `"root1::root2"`. To name the resulting variable differently you can pass a named vector: `c("name" = "pattern")` or `c("name" = "pattern(root2)")`. It's a bit intricate sorry, please see the examples.

`full` Logical scalar, defaults to `FALSE`. If `TRUE`, then all coefficients are returned, not only the aggregated coefficients.

`use_weights` Logical, default is `TRUE`. If the estimation was weighted, whether the aggregation should take into account the weights. Basically if the weights reflected frequency it should be `TRUE`.

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

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

`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.

`beta0` Deprecated function argument. Replaced by function argument 'r'.

`type` character or function. The character string specifies the type of boostrap 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)`

`bootstrap_type` Determines which wild cluster bootstrap type should be run. Options are "fnw11", which runs a "11" type wild cluster bootstrap via the algorithm outlined in "fast and wild" (Roodman et al (2019)).

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

`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.

`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.

`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", cluster.adj = "TRUE", cluster.df = "conventional"`. You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest's `fixest::ssc()` function.

`engine` Character scalar. Either "R", "R-lean" or "WildBootTests.jl". Controls if `boottest()` should run via its native R implementation or `WildBootTests.jl`. "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. The "R-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. You can set the employed algorithm globally by using the `setBoottest_engine()` function.

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

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

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

`sampling` 'dqrng' or 'standard'. If 'dqrng', the 'dqrng' package is used for random number generation (when available). If 'standard', functions from the 'stats' package are used when available. This argument is mostly a convenience to control random number generation in a wrapper package around `fwildclusterboot`, `wildrwolf`. I recommend to use the fast' option.

`...` misc function arguments

Details

Note that contrary to the SA article, here the cohort share in the sample is considered to be a perfect measure for the cohort share in the population.

Most of this function is written by Laurent Bergé and used in the fixest package published under GPL-3, https://cran.r-project.org/web/packages/fixest/index.html minor changes by Alexander Fischer

Value

A data frame with aggregated coefficients, p-values and confidence intervals.

Examples

```
## Not run:
if(requireNamespace("fixest")){
library(fixest)
data(base_stagg)
# The DiD estimation
res_sunab = feols(y ~ x1 + sunab(year_treated, year) | id + year, base_stagg)
res_sunab_3ref = feols(y ~ x1 + sunab(
 year_treated, year, ref.p = c(.F + 0:2, -1)) |
                        id + year,
                      cluster = "id",
                      base_stagg,
                      ssc = ssc(adj = FALSE, cluster.adj = FALSE))

aggregate(res_sunab, agg = "ATT")
# test ATT equivalence
boot_att <-
 boot_aggregate(
   res_sunab,
   B = 9999,
   agg = "ATT",
   clustid = "id"
 )
head(boot_att)

#'boot_agg2 <-
 boot_aggregate(
   res_sunab,
   B = 99999,
   agg = TRUE,
   ssc = boot_ssc(adj = FALSE, cluster.adj = FALSE)
 )

}

## End(Not run)
```

`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"
)
```

Arguments

`adj`	Logical scalar, defaults to TRUE. If TRUE, applies a small sample correction of (N-1) / (N-k) where N is the number of observations and k is the number of estimated coefficients excluding any fixed effects projected out in either fixest::feols() or lfe::felm().
`fixef.K`	Character scalar, equal to 'none': the fixed effects parameters are discarded when calculating k in (N-1) / (N-k).
`cluster.adj`	Logical scalar, defaults to TRUE. If TRUE, a cluster correction G/(G-1) is performed, with G the number of clusters.
`cluster.df`	Either "conventional"(the default) or "min". Controls how "G" is computed for multiway clustering if cluster.adj = TRUE. Note that the covariance matrix in the multiway clustering case is of the form V = V_1 + V_2 - V_12. If "conventional", then each summand G_i is multiplied with a small sample adjustment G_i / (G_i - 1). If "min", all summands are multiplied with the same value, min(G) / (min(G) - 1)

Value

A list with encoded info on how to form small sample corrections

Examples

```
boot_ssc(adj = TRUE, cluster.adj = TRUE)
boot_ssc(adj = TRUE, cluster.adj = TRUE, cluster.df = "min")
```

`confint.boottest` *S3 method to obtain wild cluster bootstrapped confidence intervals*

Description

S3 method to obtain wild cluster bootstrapped confidence intervals

Usage

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

Arguments

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

Value

A vector containing the boundaries of the wild cluster bootstrapped confidence interval

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
  proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
teststat(boot)
```

`find_proglang` *Check if julia or python are installed / can be found on the PATH.*

Description

Based on Mauro Lepore's great suggestion https://github.com/ropensci/software-review/issues/546#issuecomment-1416728843

Usage

```
find_proglang(lang)
```

Arguments

`lang`	which language to check. Either 'julia' or 'python'

Value

logical. TRUE if lang is found on path, FALSE if not

Examples

```
## Not run:
find_proglang(lang = "julia")

## End(Not run)
```

---

glance.boottest *S3 method to glance at objects of class boottest*

Description

S3 method to glance at objects of class boottest

Usage

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

Arguments

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

Value

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

Examples

```
## Not run:
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
generics::glance(boot)

## End(Not run)
```

glance.mboottest *S3 method to glance at objects of class boottest*

Description

S3 method to glance at objects of class boottest

Usage

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

Arguments

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

Value

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

Examples

```
## Not run:
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
mboot <- mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
)
generics::glance(mboot)

## End(Not run)
```

`mboottest` *Arbitrary Linear Hypothesis Testing for Regression Models via Wald-Tests*

Description

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

Usage

```
mboottest(object, ...)
```

Arguments

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

Value

An object of class `mboottest`.

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

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (`https://ideas.repec.org/p/qed/wpaper/1406.html`)

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. "Reworking wild bootstrap based inference for clustered errors" . No. 1315. Queen's Economics Department Working Paper, 2013.

See Also

mboottest.lm mboottest.felm mboottest.fixest

Examples

```
## Not run:
requireNamespace("clubSandwich")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
summary(wboottest)
print(wboottest)
nobs(wboottest)
pval(wboottest)
generics::tidy(wboottest)

## End(Not run)
```

---

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,
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  floattype = "Float64",
  getauxweights = 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.

`type` character or function. The character string specifies the type of boostrap 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 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. 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)

`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", cluster.adj = "TRUE", cluster.df = "conventional"`. You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest's `fixest::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.

Setting Seeds

To guarantee reproducibility, you need to set a global random seed via `set.seed()` when using

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (`https://ideas.repec.org/p/qed/wpaper/1406.html`)

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. "Reworking wild bootstrap based inference for clustered errors" . No. 1315. Queen's Economics Department Working Paper, 2013.

Examples

```
## Not run:
requireNamespace("lfe")
requireNamespace("clubSandwich")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
summary(wboottest)
print(wboottest)
nobs(wboottest)
pval(wboottest)
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

```
## S3 method for class 'fixest'
mboottest(
  object,
  clustid,
  B,
  R,
  r = rep(0, nrow(R)),
  bootcluster = "max",
  fe = NULL,
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  floattype = "Float64",
  getauxweights = 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 feols

`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.

`type` character or function. The character string specifies the type of boostrap 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 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. 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)

`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", cluster.adj = "TRUE", cluster.df = "conventional"`. You can find more details in the help file for `boot_ssc()`. The function is purposefully designed to mimic fixest's `fixest::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.

Setting Seeds

To guarantee reproducibility, you need to set a global random seed via`set.seed()`

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (`https://ideas.repec.org/p/qed/wpaper/1406.html`)

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. "Reworking wild bootstrap based inference for clustered errors" . No. 1315. Queen's Economics Department Working Paper, 2013.

Examples

```
## Not run:
requireNamespace("fwildclusterboot")
requireNamespace("clubSandwich")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
summary(wboottest)
print(wboottest)
nobs(wboottest)
pval(wboottest)
generics::tidy(wboottest)

## End(Not run)
```

---

mboottest.lm — *Fast wild cluster bootstrap inference of joint hypotheses for object of class 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

```
## S3 method for class 'lm'
mboottest(
  object,
  clustid,
  B,
  R,
  r = rep(0, nrow(R)),
  bootcluster = "max",
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  floattype = "Float64",
  getauxweights = 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.

`type` character or function. The character string specifies the type of boostrap 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 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. 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) 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", cluster.adj = "TRUE", cluster.df = "conventional". You can find more details in the help file for boot_ssc(). The function is purposefully designed to mimic fixest's fixest::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.

Setting Seeds

To guarantee reproducibility, you need to set a global random seed via set.seed()

References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (https://ideas.repec.org/p/qed/wpaper/1406.html)

Cameron, A. Colin, Jonah B. Gelbach, and Douglas L. Miller. "Bootstrap-based improvements for inference with clustered errors." The Review of Economics and Statistics 90.3 (2008): 414-427.

Cameron, A.Colin & Douglas L. Miller. "A practitioner's guide to cluster-robust inference" Journal of Human Resources (2015) doi:10.3368/jhr.50.2.317

Davidson & MacKinnon. "Wild Bootstrap Tests for IV regression" Journal of Economics and Business Statistics (2010) doi:10.1198/jbes.2009.07221

MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

MacKinnon, James G., and Matthew D. Webb. "Cluster-robust inference: A guide to empirical practice" Journal of Econometrics (2022) doi:10.1016/j.jeconom.2022.04.001

MacKinnon, James. "Wild cluster bootstrap confidence intervals." L'Actualite economique 91.1-2 (2015): 11-33.

Webb, Matthew D. "Reworking wild bootstrap based inference for clustered errors" . No. 1315. Queen's Economics Department Working Paper, 2013.

Examples

```
## Not run:
requireNamespace("clubSandwich")
requireNamespace("fwildclusterboot")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
summary(wboottest)
print(wboottest)
nobs(wboottest)
pval(wboottest)
generics::tidy(wboottest)

## End(Not run)
```

`nobs.boottest` *S3 method to obtain the effective number of observation used in* `boottest()`

Description

S3 method to obtain the effective number of observation used in `boottest()`

Usage

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

Arguments

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

Value

A scalar containing the effective number of observations used in `boottest()`

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
nobs(boot)
```

`nobs.mboottest` *S3 method to obtain the effective number of observation used in* `mboottest()`

Description

S3 method to obtain the effective number of observation used in `mboottest()`

Usage

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

Arguments

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

Value

A scalar containing the effective number of observations used in `mboottest()`

Examples

```
## Not run:
requireNamespace("clubSandwich")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
nobs(wboottest)

## End(Not run)
```

---

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

Description

Plot the bootstrap distribution of t-statistics

Usage

```
## 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

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
plot(boot)
```

print.boottest — *S3 method to print key information for objects of type* bboottest

Description

S3 method to print key information for objects of type bboottest

Usage

```
## S3 method for class 'boottest'
print(x, ..., digits = 4)
```

Arguments

x	object of type boottest
...	Further arguments passed to or from other methods.
digits	Number of rounding digits

Value

A scalar containing the effective number of observations used in mboottest

Examples

```
#' requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
  proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
print(boot)
```

print.mboottest — *S3 method to print key information for objects of type* `mboottest`

Description

S3 method to print key information for objects of type `mboottest`

Usage

```
## S3 method for class 'mboottest'
print(x, ..., digits = 4)
```

Arguments

`x`	object of type mboottest
`...`	Further arguments passed to or from other methods.
`digits`	Number of rounding digits

Value

A scalar containing the effective number of observations used in `mboottest`

Examples

```
## Not run:
requireNamespace("clubSandwich")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
print(wboottest)

## End(Not run)
```

pval — pval *is a S3 method to collect pvalues for objects of type* boottest *and* mboottest

Description

pval is a S3 method to collect pvalues for objects of type boottest and mboottest

Usage

```
pval(object, ...)
```

Arguments

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

Value

A scalar with the bootstrapped p-value.

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
pval(boot)
```

pval.boottest — *S3 method to obtain the wild cluster bootstrapped p-value of an object of type boottest*

Description

S3 method to obtain the wild cluster bootstrapped p-value of an object of type boottest

Usage

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

Arguments

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

Value

A vector containing the boundaries of the wild cluster bootstrapped p-value

Examples

```
#' requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
  proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
confint(boot)
```

`pval.mboottest` *S3 method to obtain the wild cluster bootstrapped p-value of an object of type mboottest*

Description

S3 method to obtain the wild cluster bootstrapped p-value of an object of type mboottest

Usage

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

Arguments

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

Value

A vector containing the boundaries of the wild cluster bootstrapped p-value

Examples

```
## Not run:
requireNamespace("clubSandwich")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
pval(wboottest)

## End(Not run)
```

---

setBoottest_engine *Sets the default bootstrap algo for the current R session to be run via* `boottest()` *and* `mboottest()`

Description

Sets the default bootstrap algo for the current R session to be run via `boottest()` and `mboottest()`

Usage

```
setBoottest_engine(engine)
```

Arguments

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

Value

No return value

Examples

```
## Not run:
setBoottest_engine(engine = "R")
setBoottest_engine(engine = "WildBootTests.jl")

## End(Not run)
```

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

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
summary(boot)
```

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

Examples

```
## Not run:
requireNamespace(”clubSandwich”)
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = ”group_id1”,
    B = 999,
    R = R
  )
summary(wboottest)
print(wboottest)
nobs(wboottest)
pval(wboottest)
generics::tidy(wboottest)

## End(Not run)
```

`teststat` *`teststat` is a S3 method to collect teststats for objects of type `boottest` and `mboottest`*

Description

`teststat` is a S3 method to collect teststats for objects of type `boottest` and `mboottest`

Usage

```
teststat(object, ...)
```

Arguments

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

Value

A scalar with containing the non-bootstrapped test statistic of interest

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
  proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
teststat(boot)
```

`teststat.boottest` *S3 method to obtain the non-bootstrapped t-statistic calculated via* `boottest()`

Description

S3 method to obtain the non-bootstrapped t-statistic calculated via `boottest()`

Usage

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

Arguments

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

Value

A vector containing the non-bootstrapped t-statistic calculated in `boottest()`

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
  proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
teststat(boot)
```

---

teststat.mboottest *S3 method to obtain the non-bootstrapped test statistic calculated via* `mboottest()`

Description

S3 method to obtain the non-bootstrapped test statistic calculated via `mboottest()`

Usage

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

Arguments

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

Value

A vector containing the non-bootstrapped t-statistic calculated in `mboottest()`

Examples

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

```
teststat(wboottest)

## End(Not run)
```

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(x, ...)
```

Arguments

x	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

Examples

```
requireNamespace("fwildclusterboot")
data(voters)
lm_fit <- lm(
proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
generics::tidy(boot)
```

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(x, ...)
```

Arguments

x	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

Examples

```
## Not run:
requireNamespace("clubSandwich")
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
summary(wboottest)
print(wboottest)
nobs(wboottest)
pval(wboottest)
generics::tidy(wboottest)

## End(Not run)
```

`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