

# Package ‘fwildclusterboot’

April 19, 2022

**Title** Fast Wild Cluster Bootstrap Inference for Linear Models

**Version** 0.8

**Date** 2022-04-18

**Description** Implementation of the fast algorithm for wild cluster bootstrap inference developed in Roodman et al (2019, STATA Journal) for linear regression models <[doi:10.1177/1536867X19830877](https://doi.org/10.1177/1536867X19830877)>, which makes it feasible to quickly calculate bootstrap test statistics based on a large number of bootstrap draws even for large samples. Multiway clustering, regression weights, bootstrap weights, fixed effects and subcluster bootstrapping are supported. Further, both restricted (WCR) and unrestricted (WCU) bootstrap are supported. Methods are provided for a variety of fitted models, including 'lm()', 'feols()' (from package 'fixest') and 'felm()' (from package 'lfe'). Additionally implements a heteroskedasticity-robust (HC1) wild bootstrap. Further, the package provides an R binding to 'WildBootTests.jl', which provides additional speed gains and functionality, including the 'WRE' bootstrap for instrumental variable models (based on models of type 'ivreg()' from package 'ivreg') and hypotheses with  $q > 1$ .

**URL** <https://s3alfisc.github.io/fwildclusterboot/>

**BugReports** <https://github.com/s3alfisc/fwildclusterboot/issues/>

**License** GPL-3

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

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

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.2

**LinkingTo** Rcpp,RcppArmadillo, RcppEigen

**VignetteBuilder** knitr

**Language** en-US

**SystemRequirements** Julia (>= 1.7), WildBootTests.jl (>=0.7.13)

**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** 2022-04-19 00:04:29 UTC

## R topics documented:

.onLoad	3
boottest	3
boottest.felm	4
boottest.fixest	9
boottest.ivreg	15
boottest.lm	19
boot_algo1	24
boot_algo2	25
boot_ssc	27
check_deleted_obs	28
cpp_get_nb_threads	28
create_data	28
eigenMapMatMult	29
get_seed	30
get_ssc	30
glance.boottest	31
mboottest	31
mboottest.felm	32
mboottest.fixest	36
mboottest.lm	40
plot.boottest	43
setBoottest_boot_algo	44
summary.boottest	44
summary.mboottest	45
tidy.boottest	45
tidy.mboottest	46

<code>.onLoad</code>	3
<code>to_integer</code> . . . . .	46
<code>voters</code> . . . . .	47
<b>Index</b>	<b>48</b>

---

<code>.onLoad</code>	<i>setting options for nthreads when package is loaded</i>
----------------------	--

---

**Description**

setting options for nthreads when package is loaded

**Usage**

```
.onLoad(libname, pkgname)
```

**Arguments**

<code>libname</code>	library name
<code>pkgname</code>	package name

**Value**

Changes number of threads used.

---

<code>boottest</code>	<i>Fast wild cluster bootstrap inference</i>
-----------------------	--

---

**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**

<code>object</code>	An object of type <code>lm</code> , <code>fixest</code> , <code>felm</code> or <code>ivreg</code>
<code>...</code>	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 `boot_algo = "R-lean"`), 2) the heteroskedastic wild bootstrap
  2. the wild cluster bootstrap via `boot_algo = "R"` with Mammen weights or 4) `boot_algo = "WildBootTests.jl"`
- `dqrng::dqset.seed()` when using `boot_algo = "R"` for Rademacher, Webb or Normal weights

## References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)

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.

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

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](#)

---

`boottest.felm`

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

---

## Description

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

## Usage

```
## S3 method for class 'felm'
boottest(
  object,
  param,
  B,
  clustid = NULL,
  bootcluster = "max",
```

```

fe = NULL,
conf_int = TRUE,
seed = NULL,
R = NULL,
r = 0,
beta0 = NULL,
sign_level = 0.05,
type = "rademacher",
impose_null = TRUE,
p_val_type = "two-tailed",
tol = 1e-06,
maxiter = 10,
na_omit = TRUE,
nthreads = getBoottest_nthreads(),
ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
  "conventional"),
boot_algo = getBoottest_boot_algo(),
floattype = "Float64",
maxmatsize = FALSE,
bootstrapc = FALSE,
t_boot = FALSE,
getauxweights = FALSE,
...
)

```

### Arguments

object	An object of class <code>felm</code>
param	A character vector. 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 containing the names of the cluster variables. If NULL, a heteroskedasticity-robust (HC1) wild bootstrap is run.
bootcluster	A character vector. 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 <code>boottest</code> command, the default is to cluster by the intersection of all the variables specified via the <code>clustid</code> 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 of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, <code>fe</code> 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.
seed	An integer. Allows to set a random seed. For details, see below.
R	Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as param. If NULL, a vector of ones of length param.
r	A numeric. Shifts the null hypothesis $H_0: \text{param} = r$ vs $H_1: \text{param} \neq r$
beta0	Deprecated function argument. Replaced by function argument 'r'.
sign_level	A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. sign_level = 0.05 returns 0.95% confidence intervals. By default, sign_level = 0.05.
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher distribution, if the number of replications B exceeds the number of possible draw combinations, $2^{(\text{number of clusters})}$ , then boottest() will use each possible combination once (enumeration).
impose_null	Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)
p_val_type	Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".
tol	Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. 1e-6 by default.
maxiter	Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.
na_omit	Logical. If TRUE, boottest() omits rows with missing variables in the cluster variable that have not previously been deleted when fitting the regression object (e.g. if the cluster variable was not used when fitting the regression model).
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 <code>boot_ssc</code> . Represents how the small sample adjustments are computed. The defaults are <code>adj = TRUE</code> , <code>fixef.K = "none"</code> , <code>cluster.adj = "TRUE"</code> , <code>cluster.df = "conventional"</code> . You can find more details in the help file for <code>boot_ssc()</code> . The function is purposefully designed to mimic <code>fixest</code> 's <code>ssc</code> function.
boot_algo	Character scalar. Either "R" or "WildBootTests.jl". Controls the algorithm employed by boottest. "R" is the default and implements the cluster bootstrap as in Roodman (2019). "WildBootTests.jl" executes the wild cluster bootstrap by via the WildBootTests.jl package. For it to run, Julia and WildBootTests.jl need to be installed. Check out the <code>set_up_...</code> functions The "fast and wild" algorithm is extremely fast for small number of clusters, but because it is fully vectorized, very memory-demanding. For large number of clusters and large number of

bootstrap iterations, the fast and wild algorithm becomes infeasible. If a out-of-memory error # occurs, the "lean" algorithm is a memory friendly, but less performant rcpp-armadillo based implementation of the wild cluster bootstrap. Note that if no cluster is provided, boottest() always defaults to the "lean" algorithm. Note that you can set the employed algorithm globally by using the setBoottest\_boot\_algo() function.

floattype	Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit? Only relevant when 'boot_algo = "WildBootTests.jl"'
maxmatsize	NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes. Only relevant when 'boot_algo = "WildBootTests.jl"'
bootstrapc	Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t. Only relevant when 'boot_algo = "WildBootTests.jl"'
t_boot	Logical. Should bootstrapped t-statistics be returned?
getauxweights	Logical. Whether to save auxilliary weight matrix (v)
...	Further arguments passed to or from other methods.

## Value

An object of class boottest

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

## Setting Seeds

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

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

## 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  $p_x$  cross the significance level `sign_level`.
- Feed the two pairs of  $x$  into an numerical root finding procedure and solve for the root. `boottest` currently relies on `stats::uniroot` and sets an absolute tolerance of  $1e-06$  and stops the procedure after 10 iterations.

## Standard Errors

`boottest` does not calculate standard errors.

## References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using `boottest`", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)

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.

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

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:
if (requireNamespace("lfe")) {
  library(lfe)
  data(voters)
  felm_fit <- felm(proposition_vote ~ treatment + ideology1 + log_income |
```



```

    Q1_immigration,
  data = voters
)
boot1 <- boottest(felm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
boot2 <- boottest(felm_fit,
  B = 9999,
  param = "treatment",
  clustid = c("group_id1", "group_id2")
)
boot3 <- boottest(felm_fit,
  B = 9999,
  param = "treatment",
  clustid = c("group_id1", "group_id2"),
  fe = "Q1_immigration"
)
boot4 <- boottest(felm_fit,
  B = 999,
  param = "treatment",
  clustid = c("group_id1", "group_id2"),
  fe = "Q1_immigration",
  sign_level = 0.2,
  seed = 8,
  r = 2
)
# test treatment + ideology1 = 2
boot5 <- boottest(felm_fit,
  B = 9999,
  clustid = c("group_id1", "group_id2"),
  param = c("treatment", "ideology1"),
  R = c(1, 1),
  r = 2
)
summary(boot1)
plot(boot1)
}

## End(Not run)

```

---

boottest.fixest

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


---

### Description

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

**Usage**

```

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

```

**Arguments**

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

<code>fe</code>	A character vector of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, <code>fe</code> needs to be NULL.
<code>sign_level</code>	A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. <code>sign_level = 0.05</code> returns 0.95% confidence intervals. By default, <code>sign_level = 0.05</code> .
<code>conf_int</code>	A logical vector. If TRUE, boottest computes confidence intervals by test inversion. If FALSE, only the p-value is returned.
<code>seed</code>	An integer. Allows to set a random seed. For details, see below.
<code>R</code>	Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as <code>param</code> . If NULL, a vector of ones of length <code>param</code> .
<code>r</code>	A numeric. Shifts the null hypothesis $H_0: \text{param} = r$ vs $H_1: \text{param} \neq r$
<code>beta0</code>	Deprecated function argument. Replaced by function argument 'r'.
<code>type</code>	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, <code>type</code> can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher distribution, if the number of replications <code>B</code> exceeds the number of possible draw combinations, $2^{(\text{number of clusters})}$ , then <code>boottest()</code> will use each possible combination once (enumeration).
<code>impose_null</code>	Logical. Controls if the null hypothesis is imposed on the bootstrap <code>dgp</code> or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)
<code>p_val_type</code>	Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".
<code>tol</code>	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.
<code>maxiter</code>	Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.
<code>na_omit</code>	Logical. If TRUE, <code>boottest()</code> omits rows with missing variables in the cluster variable that have not previously been deleted when fitting the regression object (e.g. if the cluster variable was not used when fitting the regression model).
<code>nthreads</code>	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.
<code>ssc</code>	An object of class <code>boot_ssc.type</code> obtained with the function <code>boot_ssc</code> . Represents how the small sample adjustments are computed. The defaults are <code>adj = TRUE</code> , <code>fixef.K = "none"</code> , <code>cluster.adj = "TRUE"</code> , <code>cluster.df = "conventional"</code> . You can find more details in the

	help file for <code>boot_ssc()</code> . The function is purposefully designed to mimic <code>fixest</code> 's <code>ssc</code> function.
<code>boot_algo</code>	Character scalar. Either "R" or "WildBootTests.jl". Controls the algorithm employed by <code>boottest()</code> . "R" is the default and implements the cluster bootstrap as in Roodman (2019). "WildBootTests.jl" executes the wild cluster bootstrap via the <code>WildBootTests.jl</code> package. For it to run, Julia and <code>WildBootTests.jl</code> need to be installed. Note that if no cluster is provided, <code>boottest()</code> always defaults to the "lean" algorithm. You can set the employed algorithm globally by using the <code>setBoottest_boot_algo()</code> function.
<code>floattype</code>	Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit? Only relevant when <code>'boot_algo = "WildBootTests.jl"</code>
<code>maxmatsize</code>	NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix ( <code>v</code> ), in gigabytes. Only relevant when <code>'boot_algo = "WildBootTests.jl"</code>
<code>bootstrapc</code>	Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t. Only relevant when <code>'boot_algo = "WildBootTests.jl"</code>
<code>t_boot</code>	Logical. Should bootstrapped t-statistics be returned?
<code>getauxweights</code>	Logical. Whether to save auxilliary weight matrix ( <code>v</code> )
<code>...</code>	Further arguments passed to or from other methods.

## Value

An object of class `boottest`

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

t_boot	All bootstrap t-statistics.
regression	The regression object used in boottest.
call	Function call of boottest.
boot_algo	The employed bootstrap algorithm.
nthreads	The number of threads employed.
internal_seed	The integer value -inherited from set.seed() - used within boottest() to set the random seed in either R or Julia. If NULL, no internal seed was created.

### Setting Seeds

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

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

### 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  $p_x$  cross the significance `sign_level`.
- Feed the two pairs of  $x$  into an numerical root finding procedure and solve for the root. boottest currently relies on `stats::uniroot` and sets an absolute tolerance of  $1e-06$  and stops the procedure after 10 iterations.

### Standard Errors

boottest does not calculate standard errors.

### References

- Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)
- 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.
- MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.
- 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:
if (requireNamespace("fixest")) {
  library(fwildclusterboot)
  library(fixest)
  data(voters)
  feols_fit <- feols(proposition_vote ~ treatment + ideology1 + log_income,
    fixef = "Q1_immigration",
    data = voters
  )
  boot1 <- boottest(feols_fit,
    B = 9999,
    param = "treatment",
    clustid = "group_id1"
  )
  boot2 <- boottest(feols_fit,
    B = 9999,
    param = "treatment",
    clustid = c("group_id1", "group_id2")
  )
  boot3 <- boottest(feols_fit,
    B = 9999,
    param = "treatment",
    clustid = c("group_id1", "group_id2"),
    fe = "Q1_immigration"
  )
  boot4 <- boottest(feols_fit,
    B = 9999,
    param = "treatment",
    clustid = c("group_id1", "group_id2"),
    fe = "Q1_immigration",
    sign_level = 0.2,
    seed = 8,
    r = 2
  )
  # test treatment + ideology1 = 2
  boot5 <- boottest(feols_fit,
    B = 9999,
    clustid = c("group_id1", "group_id2"),
    param = c("treatment", "ideology1"),
    R = c(1, 1),
    r = 2
  )
  summary(boot1)
  plot(boot1)
}

## End(Not run)

```

---

boottest.ivreg	<i>Fast wild cluster bootstrap inference for object of class lm</i>
----------------	---

---

### 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,
  seed = NULL,
  R = NULL,
  r = 0,
  sign_level = 0.05,
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  na_omit = TRUE,
  floatype = "Float64",
  getauxweights = FALSE,
  t_boot = FALSE,
  maxmatsize = NULL,
  bootstrapc = FALSE,
  liml = FALSE,
  fuller = NULL,
  kappa = NULL,
  arubin = FALSE,
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
    "conventional"),
  ...
)
```

### Arguments

object	An object of class lm
clustid	A character vector containing the names of the cluster variables

param	A character vector 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. 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 <code>clustid</code> argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.
conf_int	A logical vector. If TRUE, boottest computes confidence intervals by test inversion. If FALSE, only the p-value is returned.
seed	An integer. Allows to set a random seed. For details, see below.
R	Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as param. If NULL, a vector of ones of length param.
r	A numeric. Shifts the null hypothesis $H_0: \text{param} = r$ vs $H_1: \text{param} \neq r$
sign_level	A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. <code>sign_level = 0.05</code> returns 0.95% confidence intervals. By default, <code>sign_level = 0.05</code> .
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher and Mammen distribution, if the number of replications B exceeds the number of possible draw ombinations, $2^{\text{\#number of clusters}}$ , then <code>boottest()</code> 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.
na_omit	Logical. If TRUE, <code>boottest()</code> omits rows with missing variables in the cluster variable that have not previously been deleted when fitting the regression object (e.g. if the cluster variable was not used when fitting the regression model).
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)



t_boot	Logical. Should bootstrapped t-statistics be returned?
maxmatsize	NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes
bootstrapc	Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t
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 $\langle U+03BA \rangle$ for k-class estimation
arubin	False by default. Logical scalar. TRUE for Anderson-Rubin Test.
ssc	An object of class <code>boot_ssc</code> . type obtained with the function <code>boot_ssc</code> . Represents how the small sample adjustments are computed. The defaults are <code>adj = TRUE</code> , <code>fixef.K = "none"</code> , <code>cluster.adj = "TRUE"</code> , <code>cluster.df = "conventional"</code> . You can find more details in the help file for <code>boot_ssc()</code> . The function is purposefully designed to mimic <code>fixest</code> 's <code>ssc</code> function.
...	Further arguments passed to or from other methods.

## Value

An object of class `boot test`

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

## Setting Seeds

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

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

## References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)

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.

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

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:
library(ivreg)
library(fwildclusterboot)

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

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

## 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 the fast wild bootstrap algorithm developed in Roodman et al., 2019.

## Usage

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

## Arguments

`object`            An object of class `lm`

param	A character vector. 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 containing the names of the cluster variables. If NULL, a heteroskedasticity-robust (HC1) wild bootstrap is run.
bootcluster	A character vector. 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.
seed	An integer. Allows to set a random seed. For details, see below.
R	Hypothesis Vector giving linear combinations of coefficients. Must be either NULL or a vector of the same length as param. If NULL, a vector of ones of length param.
r	A numeric. Shifts the null hypothesis $H_0: \text{param} = r$ vs $H_1: \text{param} \neq r$
beta0	Deprecated function argument. Replaced by function argument 'r'.
sign_level	A numeric between 0 and 1 which sets the significance level of the inference procedure. E.g. sign_level = 0.05 returns 0.95% confidence intervals. By default, sign_level = 0.05.
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher distribution, if the number of replications B exceeds the number of possible draw combinations, $2^{(\text{number of clusters})}$ , then boottest() will use each possible combination once (enumeration).
impose_null	Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)
p_val_type	Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".
tol	Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. 1e-6 by default.
maxiter	Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.
na_omit	Logical. If TRUE, boottest() omits rows with missing variables in the cluster variable that have not previously been deleted when fitting the regression object (e.g. if the cluster variable was not used when fitting the regression model).

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 <code>boot_ssc.type</code> obtained with the function <code>boot_ssc</code> . Represents how the small sample adjustments are computed. The defaults are <code>adj = TRUE</code> , <code>fixef.K = "none"</code> , <code>cluster.adj = "TRUE"</code> , <code>cluster.df = "conventional"</code> . You can find more details in the help file for <code>boot_ssc()</code> . The function is purposefully designed to mimic <code>fixest</code> 's <code>ssc</code> function.
boot_algo	Character scalar. Either "R" or "WildBootTests.jl". Controls the algorithm employed by <code>boottest()</code> . "R" is the default and implements the cluster bootstrap as in Roodman (2019). "WildBootTests.jl" executes the wild cluster bootstrap via the <code>WildBootTests.jl</code> package. For it to run, Julia and <code>WildBootTests.jl</code> need to be installed. Note that if no cluster is provided, <code>boottest()</code> always defaults to the "lean" algorithm. You can set the employed algorithm globally by using the <code>setBoottest_boot_algo()</code> function.
floattype	Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit? Only relevant when <code>'boot_algo = "WildBootTests.jl"</code>
maxmatsize	NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes. Only relevant when <code>'boot_algo = "WildBootTests.jl"</code>
bootstrapc	Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t. Only relevant when <code>'boot_algo = "WildBootTests.jl"</code>
t_boot	Logical. Should bootstrapped t-statistics be returned?
getauxweights	Logical. Whether to save auxilliary weight matrix (v)
...	Further arguments passed to or from other methods.

### Value

An object of class `boot test`

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 <code>boottest</code> .
sign_level	Significance level used in <code>boottest</code> .
type	Distribution of the bootstrap weights.
impose_null	Whether the null was imposed on the bootstrap <code>dgp</code> or not.

R	The vector "R" in the null hypothesis of interest $R\beta = r$ .
r	The scalar "r" in the null hypothesis of interest $R\beta = r$ .
point_estimate	$R'\beta$ . A scalar: the constraints vector times the regression coefficients.
grid_vals	All t-statistics calculated while calculating the confidence interval.
p_grid_vals	All p-values calculated while calculating the confidence interval.
t_stat	The 'original' regression test statistics.
t_boot	All bootstrap t-statistics.
regression	The regression object used in boottest.
call	Function call of boottest.
boot_algo	The employed bootstrap algorithm.
nthreads	The number of threads employed.
internal_seed	The integer value -inherited from <code>set.seed()</code> - used within <code>boottest()</code> to set the random seed in either R or Julia. If NULL, no internal seed was created.

### Setting Seeds

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

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

### 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  $p_x$  cross the significance level `sign_level`.
- Feed the two pairs of  $x$  into an numerical root finding procedure and solve for the root. `boottest` currently relies on `stats::uniroot` and sets an absolute tolerance of  $1e-06$  and stops the procedure after 10 iterations.

### Standard Errors

`boottest` does not calculate standard errors.

## References

- Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)
- 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.
- MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.
- 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:
library(fwildclusterboot)
data(voters)
lm_fit <- lm(proposition_vote ~ treatment + ideology1 + log_income + Q1_immigration,
  data = voters
)
boot1 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = "group_id1"
)
boot2 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = c("group_id1", "group_id2")
)
boot3 <- boottest(lm_fit,
  B = 9999,
  param = "treatment",
  clustid = c("group_id1", "group_id2"),
  sign_level = 0.2,
  seed = 8,
  r = 2
)
# test treatment + ideology1 = 2
boot4 <- boottest(lm_fit,
  B = 9999,
  clustid = c("group_id1", "group_id2"),
  param = c("treatment", "ideology1"),
  R = c(1, 1),
  r = 2
)
summary(boot1)
plot(boot1)

## End(Not run)
```

boot\_algo1

*Fast wild cluster bootstrap algorithm***Description**

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

**Usage**

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

**Arguments**

preprocessed_object	A list: output of the preprocess2 function.
boot_iter	number of bootstrap iterations
point_estimate	The point estimate of the test parameter from the regression model.
impose_null	If TRUE, the null is not imposed on the bootstrap distribution. This is what Roodman et al call the "WCU" bootstrap. With impose_null = FALSE, the null is imposed ("WCR").
r	Shifts the null hypothesis.
sign_level	The significance level.
param	name of the test parameter.
p_val_type	type Type of p-value. By default "two-tailed". Other options: "equal-tailed", ">", "<"



nthreads	The number of threads. Can be: a) an integer lower than, or equal to, the maximum number of threads; b) 0: meaning all available threads will be used; c) a number strictly between 0 and 1 which represents the fraction of all threads to use. The default is to use 50\ set permanently the number of threads used within this package using the function ...
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default.
full_enumeration	Is full enumeration employed? Full enum. is used if $N_G^2 < \text{boot\_iter}$ for Mammen and Rademacher weights
small_sample_correction	The small sample correction to be applied. See ssc().
heteroskedastic	Logical - if TRUE, run a heteroskedastic. If FALSE, run wild cluster bootstrap.
seed	Integer scalar. Either set via boottest()'s seed argument or inherited from R's global seed (set via set.seed)

**Value**

A list of ...

---

boot_algo2	<i>Fast wild cluster bootstrap algorithm</i>
------------	--

---

**Description**

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

**Usage**

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

**Arguments**

preprocessed_object	A list: output of the preprocess2 function.
boot_iter	number of bootstrap iterations
point_estimate	The point estimate of the test parameter from the regression model.
impose_null	If TRUE, the null is not imposed on the bootstrap distribution. This is what Roodman et al call the "WCU" bootstrap. With impose_null = FALSE, the null is imposed ("WCR").
r	Shifts the null hypothesis.
sign_level	The significance level.
param	name of the test parameter.
p_val_type	type Type of p-value. By default "two-tailed". Other options: "equal-tailed", ">", "<"
nthreads	The number of threads. Can be: a) an integer lower than, or equal to, the maximum number of threads; b) 0: meaning all available threads will be used; c) a number strictly between 0 and 1 which represents the fraction of all threads to use. The default is to use 50\ set permanently the number of threads used within this package using the function ...
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default.
full_enumeration	Is full enumeration employed? Full enum. is used if $N_G^2 < \text{boot\_iter}$ for Mammen and Rademacher weights
small_sample_correction	The small sample correction to be applied. See ssc().
conf_int	Logical. Should confidence intervals be calculated (by test inversion)?
maxiter	Integer. Maximum number of iterations used in the root finding procedure to find the confidence interval. 10 by default.
tol	Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. 1e-6 by default.

**Value**

A list of ...

---

boot_ssc	<i>set the small sample correction factor applied in boottest()</i>
----------	---

---

## 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

<code>adj</code>	Logical scalar, defaults to <code>TRUE</code> . If <code>TRUE</code> , 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 <code>fixest::feols()</code> or <code>lfe::felm()</code> .
<code>fixef.K</code>	Character scalar, equal to <code>'none'</code> : the fixed effects parameters are discarded when calculating $k$ in $(N-1) / (N-k)$ .
<code>cluster.adj</code>	Logical scalar, defaults to <code>TRUE</code> . If <code>TRUE</code> , a cluster correction $G/(G-1)$ is performed, with $G$ the number of clusters.
<code>cluster.df</code>	Either <code>"conventional"</code> (the default) or <code>"min"</code> . Controls how $G$ is computed for multiway clustering if <code>cluster.adj = TRUE</code> . Note that the covariance matrix in the multiway clustering case is of the form $V = V_1 + V_2 - V_{12}$ . If <code>"conventional"</code> , then each summand $G_i$ is multiplied with a small sample adjustment $G_i / (G_i - 1)$ . If <code>"min"</code> , all summands are multiplied with the same value, $\min(G) / (\min(G) - 1)$

## Examples

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

---

check_deleted_obs	<i>Function checks if addition of new clustering variable leads to deletion of rows</i>
-------------------	---

---

**Description**

Function checks if addition of new clustering variable leads to deletion of rows

**Usage**

```
check_deleted_obs(N_model, N, na_omit)
```

**Arguments**

N_model	the number of observations in the original model
N	number of observations
na_omit	Logical - if TRUE, deletion of rows is enabled

---

cpp_get_nb_threads	<i>Get maximum number of threads on hardware for open mp support</i>
--------------------	--

---

**Description**

Get maximum number of threads on hardware for open mp support

**Usage**

```
cpp_get_nb_threads()
```

---

create_data	<i>Simulate Data</i>
-------------	----------------------

---

**Description**

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

**Usage**

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

**Arguments**

N	number of observations
N_G1	A scalar. number of clusters for clustering variable 1
icc1	A scalar between 0 and 1. intra-cluster correlation for clustering variable 1
N_G2	A scalar. number of clusters for clustering variable 2
icc2	A scalar between 0 and 1. intra-cluster correlation for clustering variable 2
numb_fe1	A scalar. Number of fixed effect for first factor variable
numb_fe2	A scalar. Number of fixed effect for second factor variable
seed	An integer. Set the random seed
weights	Possible regression weights to be used in estimation

**Value**

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

---

eigenMapMatMult      *Matrix Multiplication via Eigen*

---

**Description**

Matrix Multiplication via Eigen

**Usage**

```
eigenMapMatMult(A, B, nthreads)
```

**Arguments**

A	A matrix.
B	A matrix.
nthreads	Integer. Number of threads to use for matrix multiplication.

**Value**

A matrix

---

get_seed	<i>creates an integer based on the global random seed set via set.seed() for using set.seed() for controlling rcpp's seed, see this blog post <a href="http://rorynolan.rbind.io/2018/09/30/rcsetseed/">http://rorynolan.rbind.io/2018/09/30/rcsetseed/</a></i>
----------	---

---

**Description**

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

**Usage**

```
get_seed()
```

---

get_ssc	<i>Compute small sample adjustment factors</i>
---------	--

---

**Description**

Compute small sample adjustment factors

**Usage**

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

**Arguments**

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

**Value**

A small sample adjustment factor

---

glance.boottest	<i>S3 method to glance at objects of class boottest</i>
-----------------	---

---

**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

---

mboottest	<i>Arbitrary Linear Hypothesis Testing for Regression Models via Wald-Tests</i>
-----------	---

---

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

## References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)

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.

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

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](#)

---

<code>mboottest.felm</code>	<i>Fast wild cluster bootstrap inference for joint hypotheses for object of class felm</i>
-----------------------------	--

---

## Description

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

## Usage

```
## S3 method for class 'felm'
mboottest(
  object,
  clustid,
  B,
  R,
```



```

    r = rep(0, nrow(R)),
    bootcluster = "max",
    fe = NULL,
    seed = NULL,
    type = "rademacher",
    impose_null = TRUE,
    p_val_type = "two-tailed",
    tol = 1e-06,
    na_omit = TRUE,
    floattype = "Float64",
    getauxweights = FALSE,
    teststat_boot = FALSE,
    maxmatsize = NULL,
    bootstrapc = FALSE,
    ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
      "conventional"),
    ...
  )

```

### Arguments

object	An object of class <code>felm</code>
clustid	A character vector 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. 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 <code>boottest</code> command, the default is to cluster by the intersection of all the variables specified via the <code>clustid</code> 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 of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, fe needs to be NULL.
seed	An integer. Allows to set a random seed. For details, see below.
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively,

	type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher and Mammen distribution, if the number of replications B exceeds the number of possible draw ombinations, $2^{(\text{number of clusters})}$ , then <code>boottest()</code> will use each possible combination once (enumeration).
<code>impose_null</code>	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)
<code>p_val_type</code>	Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".
<code>tol</code>	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.
<code>na_omit</code>	Logical. If TRUE, <code>boottest()</code> omits rows with missing variables in the cluster variable that have not previously been deleted when fitting the regression object (e.g. if the cluster variable was not used when fitting the regression model).
<code>floattype</code>	Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit?
<code>getauxweights</code>	Logical. FALSE by default. Whether to save auxilliary weight matrix (v)
<code>teststat_boot</code>	Logical. Should bootstrapped test statistics be returned?
<code>maxmatsize</code>	NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes
<code>bootstrapc</code>	Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t
<code>ssc</code>	An object of class <code>boot_ssc</code> . type obtained with the function <code>boot_ssc</code> . Represents how the small sample adjustments are computed. The defaults are <code>adj = TRUE</code> , <code>fixef.K = "none"</code> , <code>cluster.adj = "TRUE"</code> , <code>cluster.df = "conventional"</code> . You can find more details in the help file for <code>boot_ssc()</code> . The function is purposefully designed to mimic <code>fixest</code> 's <code>ssc</code> function.
<code>...</code>	Further arguments passed to or from other methods.

## Value

An object of class `mboottest`

<code>p_val</code>	The bootstrap p-value.
<code>N</code>	Sample size. Might differ from the regression sample size if the cluster variables contain NA values.
<code>boot_iter</code>	Number of Bootstrap Iterations.
<code>clustid</code>	Names of the cluster Variables.
<code>N_G</code>	Dimension of the cluster variables as used in <code>boottest</code> .
<code>sign_level</code>	Significance level used in <code>boottest</code> .
<code>type</code>	Distribution of the bootstrap weights.
<code>impose_null</code>	Whether the null was imposed on the bootstrap dgp or not.

R	The vector "R" in the null hypothesis of interest $R\beta = r$ .
r	The scalar "r" in the null hypothesis of interest $R\beta = r$ .
point_estimate	$R'\beta$ . A scalar: the constraints vector times the regression coefficients.
teststat_stat	The 'original' regression test statistics.
teststat_boot	All bootstrap t-statistics.
regression	The regression object used in boottest.
call	Function call of boottest.
boot_algo	The employed bootstrap algorithm.
nthreads	The number of threads employed.
internal_seed	The integer value -inherited from <code>set.seed()</code> - used within <code>boottest()</code> to set the random seed in either R or Julia. If NULL, no internal seed was created.

### Setting Seeds

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

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

### References

- Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using boottest", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)
- 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.
- MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." *The Econometrics Journal* 21.2 (2018): 114-135.
- 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:
library(lfe)
library(clusterSandwich)
R <- clusterSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
```

```

    clustid = "group_id1",
    B = 999,
    R = R
  )
  generics::tidy(wboottest)

## End(Not run)

```

---

mboottest.fixest	<i>Fast wild cluster bootstrap inference for joint hypotheses for object of class fixest</i>
------------------	--

---

## 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,
  seed = NULL,
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  na_omit = TRUE,
  floatype = "Float64",
  getauxweights = FALSE,
  teststat_boot = FALSE,
  maxmatsize = NULL,
  bootstrapc = FALSE,
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
    "conventional"),
  ...
)

```

**Arguments**

object	An object of class feols
clustid	A character vector 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. 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 of length one which contains the name of the fixed effect to be projected out in the bootstrap. Note: if regression weights are used, fe needs to be NULL.
seed	An integer. Allows to set a random seed. For details, see below.
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher and Mammen distribution, if the number of replications B exceeds the number of possible draw combinations, $2^{\text{number of clusters}}$ , then boottest() will use each possible combination once (enumeration).
impose_null	Logical. Controls if the null hypothesis is imposed on the bootstrap dgp or not. Null imposed (WCR) by default. If FALSE, the null is not imposed (WCU)
p_val_type	Character vector of length 1. Type of p-value. By default "two-tailed". Other options include "equal-tailed", ">" and "<".
tol	Numeric vector of length 1. The desired accuracy (convergence tolerance) used in the root finding procedure to find the confidence interval. Relative tolerance of 1e-6 by default.
na_omit	Logical. If TRUE, boottest() omits rows with missing variables in the cluster variable that have not previously been deleted when fitting the regression object (e.g. if the cluster variable was not used when fitting the regression model).
floattype	Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit?

getauxweights	Logical. FALSE by default. Whether to save auxilliary weight matrix (v)
teststat_boot	Logical. Should bootstrapped test statistics be returned?
maxmatsize	NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes
bootstrapc	Logical scalar, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t
ssc	An object of class <code>boot_ssc.type</code> obtained with the function <code>boot_ssc</code> . Represents how the small sample adjustments are computed. The defaults are <code>adj = TRUE</code> , <code>fixef.K = "none"</code> , <code>cluster.adj = "TRUE"</code> , <code>cluster.df = "conventional"</code> . You can find more details in the help file for <code>boot_ssc()</code> . The function is purposefully designed to mimic <code>fixest</code> 's <code>ssc</code> function.
...	Further arguments passed to or from other methods.

### Value

An object of class `mboottest`

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

## Setting Seeds

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

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

## References

Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using `boottest`", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)

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.

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

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:
library(fwildclusterboot)
library(clubSandwich)
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
generics::tidy(wboottest)

## End(Not run)
```

---

mboottest.lm	<i>Fast wild cluster bootstrap inference of joint hypotheses for object of class lm</i>
--------------	---

---

## 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",
  seed = NULL,
  type = "rademacher",
  impose_null = TRUE,
  p_val_type = "two-tailed",
  tol = 1e-06,
  na_omit = TRUE,
  floattype = "Float64",
  getauxweights = FALSE,
  teststat_boot = FALSE,
  maxmatsize = NULL,
  bootstrapc = FALSE,
  ssc = boot_ssc(adj = TRUE, fixef.K = "none", cluster.adj = TRUE, cluster.df =
    "conventional"),
  ...
)
```

## Arguments

object	An object of class lm
clustid	A character vector 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. 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 <code>clustid</code> argument, even though that is not necessarily recommended (see the paper by Roodman et al cited below, section 4.2). Other options include "min", where bootstrapping is clustered by the cluster variable with the fewest clusters. Further, the subcluster bootstrap (MacKinnon & Webb, 2018) is supported - see the vignette("fwildclusterboot", package = "fwildclusterboot") for details.
seed	An integer. Allows to set a random seed. For details, see below.
type	character or function. The character string specifies the type of bootstrap to use: One of "rademacher", "mammen", "norm", "gamma" and "webb". Alternatively, type can be a function(n) for drawing wild bootstrap factors. "rademacher" by default. For the Rademacher and Mammen distribution, if the number of replications B exceeds the number of possible draw ombinations, $2^{(\text{number of clusters})}$ , then <code>boottest()</code> 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.
na_omit	Logical. If TRUE, <code>boottest()</code> omits rows with missing variables in the cluster variable that have not previously been deleted when fitting the regression object (e.g. if the cluster variable was not used when fitting the regression model).
floattype	Float64 by default. Other option: Float32. Should floating point numbers in Julia be represented as 32 or 64 bit?
getauxweights	Logical. FALSE by default. Whether to save auxilliary weight matrix (v)
teststat_boot	Logical. Should bootstrapped test statistics be returned?
maxmatsize	NULL by default = no limit. Else numeric scalar to set the maximum size of auxilliary weight matrix (v), in gigabytes
bootstrapc	Logical, FALSE by default. TRUE to request bootstrap-c instead of bootstrap-t
ssc	An object of class <code>boot_ssc</code> . type obtained with the function <code>boot_ssc</code> . Represents how the small sample adjustments are computed. The defaults are <code>adj = TRUE</code> , <code>fixef.K = "none"</code> , <code>cluster.adj = "TRUE"</code> , <code>cluster.df = "conventional"</code> . You can find more details in the help file for <code>boot_ssc()</code> . The function is purposefully designed to mimic <code>fixest</code> 's <code>ssc</code> function.
...	Further arguments passed to or from other methods.

**Value**

An object of class `mboottest`

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

**Setting Seeds**

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

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

**References**

- Roodman et al., 2019, "Fast and wild: Bootstrap inference in STATA using `boottest`", The STATA Journal. (<https://journals.sagepub.com/doi/full/10.1177/1536867X19830877>)
- 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.
- MacKinnon, James G., and Matthew D. Webb. "The wild bootstrap for few (treated) clusters." The Econometrics Journal 21.2 (2018): 114-135.

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:
library(cclubSandwich)
R <- clubSandwich::constrain_zero(2:3, coef(lm_fit))
wboottest <-
  mboottest(
    object = lm_fit,
    clustid = "group_id1",
    B = 999,
    R = R
  )
generics::tidy(wboottest)

## End(Not run)
```

---

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

---

setBoottest\_boot\_algo *Sets the bootstrap algo to be run via boottest() and waldboottest()*

---

**Description**

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

**Usage**

```
setBoottest_boot_algo(boot_algo)
```

**Arguments**

boot\_algo      Character scalar. Either 'R' or 'WildBootTests.jl'. Default is 'R'

**Value**

No return value

---

summary.boottest      *S3 method to summarize objects of class boottest*

---

**Description**

S3 method to summarize objects of class boottest

**Usage**

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

**Arguments**

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

**Value**

Returns result summaries for objects of type boottest

---

summary.mboottest      *S3 method to summarize objects of class mboottest*

---

**Description**

S3 method to summarize objects of class mboottest

**Usage**

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

**Arguments**

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

**Value**

Returns result summaries for objects of type mboottest

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

tidy.mboottest	<i>S3 method to summarize objects of class mboottest into tidy data.frame</i>
----------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

to_integer	<i>Transform vectors of all types safely to integer vectors</i>
------------	---

---

**Description**

Transform vectors of all types safely to integer vectors

**Usage**

```
to_integer(vec)
```

**Arguments**

vec	A vector
-----	----------

**Value**

An integer vector

---

voters	<i>Random example data set</i>
--------	--------------------------------

---

**Description**

Random example data set

**Usage**

```
data(voters)
```

**Format**

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

**Examples**

```
data(voters)
```

# Index

## \* datasets

- voters, [47](#)
- .onLoad, [3](#)
  
- boot\_algo1, [24](#)
- boot\_algo2, [25](#)
- boot\_ssc, [6](#), [11](#), [17](#), [21](#), [27](#), [34](#), [38](#), [41](#)
- boottest, [3](#)
- boottest.felm, [4](#), [4](#)
- boottest.fixest, [4](#), [9](#)
- boottest.ivreg, [4](#), [15](#)
- boottest.lm, [4](#), [19](#)
  
- check\_deleted\_obs, [28](#)
- cpp\_get\_nb\_threads, [28](#)
- create\_data, [28](#)
  
- eigenMapMatMult, [29](#)
  
- get\_seed, [30](#)
- get\_ssc, [30](#)
- glance.boottest, [31](#)
  
- mboottest, [31](#)
- mboottest.felm, [32](#), [32](#)
- mboottest.fixest, [32](#), [36](#)
- mboottest.lm, [32](#), [40](#)
  
- plot.boottest, [43](#)
  
- setBoottest\_boot\_algo, [44](#)
- ssc, [6](#), [12](#), [17](#), [21](#), [34](#), [38](#), [41](#)
- summary.boottest, [44](#)
- summary.mboottest, [45](#)
  
- tidy.boottest, [45](#)
- tidy.mboottest, [46](#)
- to\_integer, [46](#)
  
- voters, [47](#)