Title

stata.com

wildbootstrap — Wild cluster bootstrap inference

Description

`wildbootstrap` performs wild cluster bootstrap (WCB) inference for linear hypotheses about parameters in linear regression models. These hypotheses can be simple or composite. When the assumptions required for the consistency of the cluster–robust variance estimator (CRVE) do not hold, the WCB is a good alternative.

Quick start

Estimate the WCB p-value and confidence interval (CI) for the coefficient on `x1` in a linear regression of `y` on `x1` with clusters identified in `cvar`

```
wildbootstrap regress y x1, cluster(cvar)
```

Same as above, but test whether the coefficients on `x1` and `x2` are equal in a regression of `y` on `x1` and `x2`

```
wildbootstrap regress y x1 x2, cluster(cvar) test(x1 = x2)
```

Menu

Statistics > Resampling > Wild cluster bootstrap

Syntax

```
wildbootstrap estimator depvar [indepvars] [if] [in] [weight] [, options]
```

estimator	Description
`regress`	linear regression
`areg`	linear regression with a large indicator-variable set
`xtreg`	fixed-effects linear models

options	Description
Main	
`noconstant`	suppress constant term; available only with estimator `regress`
`hascons`	has user-supplied constant; available only with estimator `regress`
`absorb(`*varname*`)`	categorical variable to be absorbed; required by and available only with estimator `areg`
Statistics	
`coefficients(`*coeflist*`)`	perform inference for a subset of coefficients
`test(`*testspec*`)`	specify linear test parameters
Bootstrap	
`cluster(`*clustvar*`)`	specify variable identifying clusters; required for estimators `regress` and `areg`
`ptype(`*ptype*`)`	specify the p-value type; may be `equal` (the default) or `symmetric`
`errorweight(`*wcbwtype*`)`	specify WCB weight; default is `errorweight(rademacher)`
`reps(#)`	set number of bootstrap repetitions; default is `reps(1000)`
`rseed(#|`*statecode*`)`	set random-number seed to # or *statecode*
`blocksize(#)`	set bootstrap repetition block size; default is min(`reps(#)`,1000)
`cistop(largest|first)`	specify stopping rule for CI computation
Reporting	
`level(#)`	set confidence level; default is `level(95)`
display_options	control column formats and display of CIs

indepvars and *coeflist* may contain factor variables; see [U] **11.4.3 Factor variables**.

depvar, *indepvars*, and *coeflist* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Any weight that is allowed by the estimator is allowed; see [U] **11.1.6 weight**.

Options

Main

`noconstant` and `hascons`; see [R] **regress**. These options may be specified only when estimator `regress` is specified.

`absorb(`*varname*`)`; see [R] **areg**. This option must be specified when estimator `areg` is specified.

Statistics

`coefficients(`*coeflist*`)` performs an inference for a subset of coefficients. It reports the bootstrap *p*-value for a test of the subset of coefficients, *coeflist*, equal to 0 and the bootstrap CI. The default is to perform inference for all variables specified in *indepvars*. `coefficients()` may not be specified with `test()`.

`test(`*testspec*`)` specifies a linear test. You may also specify multiple linear tests by using `test((`*testspec1*`) (`*testspec2*`) ...)`. The test specification must be consistent with specifying a linear constraint. See [R] **test** and [R] **constraint**.

Bootstrap

`cluster(`*clustvar*`)` specifies the variable identifying the cluster groups. `cluster()` is required with estimators `regress` and `areg`. With estimator `xtreg`, *clustvar* defaults to the `xtset` *panelvar*.

`ptype(`*ptype*`)` specifies the *p*-value criterion: symmetric (`symmetric`) or equal tailed (`equal`). The default is `ptype(equal)`. See *Methods and formulas* for more details.

`errorweight(`*wcbwtype*`)` specifies the type of wild weight. *wcbwtype* may be one of the following:

wcbwtype	Description
`rademacher`	two-point distribution assigns values 1 and -1 with equal probability; the default
`mammen`	two-point distribution assigns value $1-\phi$ with probability $\phi/\sqrt{5}$ and value ϕ otherwise, where $\phi=(1+\sqrt{5})/2$
`webb`	six-point distribution assigns probability of $1/6$ to the points $\pm\sqrt{1/2}$, ±1, and $\pm\sqrt{3/2}$
`normal`	standard normal distribution
`gamma`	gamma distribution with shape parameter 4 and scale parameter $1/2$ centered on its mean of 2

`reps(#)` sets the number of repetitions for the bootstrap. The default is `reps(1000)`. For the `ptype(equal)` option, the values of `level()` and `reps()` should be chosen so that $\alpha/2\times$`reps(#)` is an integer, where $\alpha=(100-$`level(#)`$)/100$. For the `ptype(symmetric)` option, such values should be chosen so that $\alpha\times$`reps(#)` is an integer. When the product is not an integer, the number of repetitions is increased so that it is. Integer values improve the search efficiency of the `wildbootstrap` algorithm.

`rseed(#|`*statecode*`)` sets the random-number seed to # or *statecode*. See [R] **set seed**.

`blocksize(#)` sets the bootstrap block size. This is an integer less than or equal to `reps(#)` and is used to reduce the amount of memory the bootstrap computation will consume. The default is min(`reps(#)`,1000).

`cistop(largest|first)` specifies the stopping rule for the CI computation. The bootstrap distribution is a step function, so for each bound, there is an interval of values that meet the CI level criterion. `cistop(largest)`, the default, specifies that the largest value within the interval be selected. `cistop(first)` specifies that the first value the algorithm finds within the interval be selected; therefore, specifying `cistop(first)` will result in faster CI computation. The `cistop()` option may not be combined with the `noci` option.

Reporting

`level(#)`; see [R] **Estimation options**. The `level()` option will not work on replay because CIs are based on estimator-specific enumerations. To change the confidence level, you must refit the model.

display_options: noci, cformat(%*fmt*), pformat(%*fmt*), and sformat(%*fmt*); see [R] **Estimation options**. The noci option may not be combined with the cistop() option.

Remarks and examples

stata.com

wildbootstrap implements the WCB, which was proposed by Cameron, Gelbach, and Miller (2008). It is an extension of the original wild bootstrap procedure proposed by Wu (1986), which was designed to work well for models with heteroskedasticity, to the case of cluster-level correlation. The wild bootstrap has proven to work well in cases where cluster–robust standard errors do not perform well. A good discussion of the methodology can be found in Cameron and Miller (2015), MacKinnon (2019), MacKinnon and Webb (2018), and MacKinnon, Nielsen, and Webb (2023), and the references therein. Specifically, the WCB is a good inference tool when one or more of the CRVE t-statistic consistency assumptions are violated. MacKinnon and Webb list the assumptions as follows:

1. The number of clusters goes to infinity.
2. The within-cluster error correlations are the same for all clusters.
3. Each cluster contains an equal number of observations.

Below, we illustrate how to use wildbootstrap; however, note that alternatives exist in the literature to address the inference problems noted above. For example, the Bell and McCaffrey (2002) t-statistic degrees-of-freedom correction is an alternative to wildbootstrap when at least one of the above assumptions is violated. The degrees-of-freedom correction is computed with option vce(hc2 *clustvar*, dfadjust) for regress, areg, and xtreg, fe.

▷ Example 1: Simple regression

Say we are interested in the effect of tenure on wages for a panel of individuals sampled from 2013 to 2016. We would like to use wild bootstrap CIs clustering at the personid level. For reproducibility, we set the seed by using option rseed().

```
. use https://www.stata-press.com/data/r18/wagework
(Wages for 20 to 77 year olds, 2013-2016)
. wildbootstrap regress wage tenure, cluster(personid) rseed(12345)
Performing 1,000 replications for p-value for tenure = 0 ...
Computing confidence interval for tenure
  Lower bound: .........10.........20. done (21)
  Upper bound: .........10..... done (17)
Wild cluster bootstrap                          Number of obs      = 1,928
Linear regression                               Number of clusters =   589
                                                Cluster size:
Cluster variable: personid                                  min =     1
Error weight: Rademacher                                    avg =   3.3
                                                            max =     4
```

wage	Estimate	t	p-value	[95% conf.	interval]
constraint					
tenure = 0	.7807403	27.19	0.000	.7209754	.8368386

The iteration log indicates the number of iterations used to compute the lower and upper bound for the CIs. In *Methods and formulas* below, we discuss how these bounds are computed. Notably, there is a separate optimization procedure used to compute each one of these bounds.

The table header also tells us the error weights used for the sampling algorithm of the wild bootstrap, which, by default, are Rademacher weights. See the `errorweight()` option for more details about error weights.

The column header labeled `p-value` indicates that the t-statistic equal-tailed p-value has been computed. The `ptype(equal)` option is the default. Alternatively, the symmetric p-value is computed when the `ptype(symmetric)` option is specified and is identified with the column header of `P>|t|`.

```
. wildbootstrap regress wage tenure, cluster(personid) rseed(12345)
> ptype(symmetric)
Performing 1,000 replications for p-value for tenure = 0 ...
Computing confidence interval for tenure
  Lower bound: .........10.........20.. done (22)
  Upper bound: .........10..... done (15)
Wild cluster bootstrap                          Number of obs      = 1,928
Linear regression                               Number of clusters =   589
                                                Cluster size:
Cluster variable: personid                                  min =     1
Error weight: Rademacher                                    avg =   3.3
                                                            max =     4
```

wage	Estimate	t	P>|t|	[95% conf.	interval]
constraint					
tenure = 0	.7807403	27.19	0.000	.7240502	.8399896

We can always compare the CIs from the wild bootstrap with what we would have obtained using the underlying command, in this case, `regress`.

```
. regress
Linear regression                               Number of obs     =      1,928
                                                F(1, 588)         =     739.36
                                                Prob > F          =     0.0000
                                                R-squared         =     0.4212
                                                Root MSE          =     3.5097
                                (Std. err. adjusted for 589 clusters in personid)
```

wage	Coefficient	Robust std. err.	t	P>|t|	[95% conf.	interval]
tenure	.7807403	.028713	27.19	0.000	.7243477	.8371328
_cons	20.89884	.2135686	97.86	0.000	20.47939	21.31829

Note that typing this command will replace the return matrix `r(table)`. Observe that the t statistics are the same in both tables because `regress` and `wildbootstrap` use the same CRVE, but the p-values and CIs may vary between tables.

Similarly, you can redisplay the wildbootstrap table by typing `wildbootstrap`, which may be abbreviated as `wildboot`.

◁

▷ Example 2: Small number of clusters with wildly varying cluster sizes

As in example 1, we would like to see the effect of tenure on wages; in this case, however, we would like to cluster at the industry level. Here, for year 1988, we use a wage dataset with only 12 clusters, for which cluster sizes vary wildly from 4 to 817, violating the CRVE t-statistic consistency assumptions 1 and 3 outlined previously.

```
. use https://www.stata-press.com/data/r18/nlsw88
(NLSW, 1988 extract)
. wildbootstrap regress wage tenure, cluster(industry) rseed(12345)
Performing 1,000 replications for p-value for tenure = 0 ...
Computing confidence interval for tenure
  Lower bound: .........10.........20...... done (26)
  Upper bound: .........10.........20.... done (24)
Wild cluster bootstrap                          Number of obs      = 2,217
Linear regression                               Number of clusters =    12
                                                Cluster size:
Cluster variable: industry                                 min =     4
Error weight: Rademacher                                   avg = 184.8
                                                           max =   817
```

wage	Estimate	t	p-value	[95% conf.	interval]
constraint					
tenure = 0	.1830716	6.95	0.000	.1274023	.3258156

◁

▷ Example 3: Small number of clusters with more regressors

Continuing with example 2, we look at a more complex and perhaps more realistic example. We add to the model explanatory variables for total work experience, `ttl_exp`; a college graduate indicator, `collgrad`; and an indicator for union membership, `union`.

```
. wildbootstrap regress wage c.tenure##c.ttl_exp ib0.collgrad ib0.union,
> cluster(industry) rseed(12345)
Performing 1,000 replications for p-value for tenure = 0 ...
Computing confidence interval for tenure
  Lower bound: .........10.........20.........30.... done (34)
  Upper bound: .........10.........20.... done (24)
Performing 1,000 replications for p-value for ttl_exp = 0 ...
Computing confidence interval for ttl_exp
  Lower bound: .........10.........20..... done (25)
  Upper bound: .........10......... done (19)
Performing 1,000 replications for p-value for c.tenure#c.ttl_exp = 0 ...
Computing confidence interval for c.tenure#c.ttl_exp
  Lower bound: .........10.........20... done (23)
  Upper bound: .........10.........20....... done (27)
note: upper-bound CI achieved 1-F(-3.65e-04) = 0.0240, but target is 1-F(x) =
      .025.
note: the sorted bootstrap t statistics have at least two tied values
      adjacent to the t statistic under the null; this prevents the CI bound
      from converging to the target.
Performing 1,000 replications for p-value for 1.collgrad = 0 ...
Computing confidence interval for 1.collgrad
  Lower bound: .........10.........20.........30.. done (32)
note: lower-bound CI achieved F(1.66) = 0.0240, but target is F(x) = .025.
note: at least one bootstrap t statistic matches the t statistic under the
      null; this prevents the CI bound from converging to the target.
  Upper bound: .........10.........20...... done (26)
Performing 1,000 replications for p-value for 1.union = 0 ...
Computing confidence interval for 1.union
  Lower bound: .........10.........20.........30 done (30)
  Upper bound: .........10.........20..... done (25)
```

```
Wild cluster bootstrap                          Number of obs      = 1,855
Linear regression                               Number of clusters =    12
                                                Cluster size:
Cluster variable: industry                                 min =     2
Error weight: Rademacher                                   avg = 154.6
                                                           max =   717
```

wage	Estimate	t	p-value	[95% conf.	interval]
constraints					
tenure = 0	.204166	2.81	0.026	.0729495	.4699121
ttl_exp = 0	.3025249	11.72	0.004	.2566267	.4110114
c.tenure#c.ttl_exp = 0	-.0097942	-2.76	0.046	-.022061	-.0003651
1.collgrad = 0	3.077377	7.59	0.034	1.662277	5.218396
1.union = 0	.9114564	2.31	0.040	.1060322	2.470924

We did not specify the `coefficients()` option, so `wildbootstrap` computes p-values and CIs for all coefficients in the model, with the exception of the constant term, using the default `ptype(equal)` option to compute the equal-tailed p-values.

The iteration log states that the CI-target tail values were not achieved for two of the reported constraints. This is because the sorted vector of bootstrap t statistics can have ties due to computer finite numeric precision and the finite number of draws for the Rademacher, Mammen, and Webb distributions. When ties occur with the t statistic under the null, or those adjacent to it, and the desired CI tail area cannot be achieved, `wildbootstrap` will choose a bound that results in a smaller tail area. This is the case when `wildbootstrap` searches for the `c.tenure#c.ttl_exp` CI upper bound and the `1.collgrad` CI lower bound.

Numerical computations will contain a roundoff error; `wildbootstrap` retains 13 digits on the mantissa of the t statistics before making comparisons.

◁

▷ Example 4: Linear regression with an indicator-variable set

We would like to include an individual's occupation code (`occupation`) as an additional control in the regression to avoid potential omitted-variable bias. We keep the other control variables we introduced in example 3. For this analysis, we use the estimator `areg` with the `absorb(occupation)` option. We use the `test()` option to test linear combinations of regression estimates.

```
. wildbootstrap areg wage c.tenure##c.ttl_exp ib0.collgrad ib0.union,
> absorb(occupation) cluster(industry) rseed(12345)
> test((tenure=ttl_exp) (1.collgrad-1.union=1)) reps(1250)
note: for equal-tailed 95% CI, better performance is obtained when
      .025*reps() is an integer.
note: setting repetitions to 1,280.

Performing 1,280 replications for p-value for constraint
  tenure - ttl_exp = 0 ...
Computing confidence interval for tenure - ttl_exp
  Lower bound: .........10.........20... done (23)
  Upper bound: .........10.........20..... done (25)
note: upper-bound CI achieved 1-F(0.15) = 0.0242, but target is 1-F(x) =
      .025.
note: the sorted bootstrap t statistics have at least two tied values
      adjacent to the t statistic under the null; this prevents the CI bound
      from converging to the target.

Performing 1,280 replications for p-value for constraint
  1.collgrad - 1.union = 1 ...
```

```
Computing confidence interval for 1.collgrad - 1.union
  Lower bound: .........10.........20...... done (26)
  Upper bound: .........10.........20....... done (27)
note: upper-bound CI achieved 1-F(3.91) = 0.0242, but target is 1-F(x) =
      .025.
note: the sorted bootstrap t statistics have at least two tied values
      adjacent to the t statistic under the null; this prevents the CI bound
      from converging to the target.
Wild cluster bootstrap                              Number of obs      = 1,851
Linear regression, absorbing indicators             Number of clusters =    12
                                                    Cluster size:
Cluster variable: industry                                      min =     2
Error weight: Rademacher                                        avg = 154.2
                                                                max =   717
```

wage	Estimate	t	p-value	[95% conf.	interval]
constraints					
tenure - ttl_exp = 0	-.0391243	-0.71	0.517	-.1915527	.1495055
1.collgrad - 1.union = 1	1.356332	0.77	0.395	-.1882941	3.913359

The `test()` option allows us to specify a linear combination of the regression coefficients that we would like to test. The syntax is the same as specifying linear constraints, `constraint`, or linear hypothesis tests, `test`. In this example, we are inquiring how probable it would be for `tenure` = `ttl_exp` and `1.collgrad` − `1.union` = 1.

The `reps()` option allows us to set the number of bootstrap repetitions. The default is 1,000. Because the Rademacher and Mammen distributions have two possible realizations, the maximum number of possible bootstrap samples is 2^G, where G is the number of clusters. The Webb distribution, on the other hand, has six outcomes and therefore 6^G possible combinations.

In this example, we intentionally specify a number of bootstrap repetitions that results in a noninteger product, `reps()` $\times\ \alpha/2$: `reps()` $\times\ (100 -$ `level()`$)/200 = 1250 \times 0.025 = 31.25$. In this case, we lose efficiency in searching for the CI bounds, so `wildbootstrap` adjusts the number of repetitions to 1,280; when we specify `reps(1250)` with `level(95)`, `wildbootstrap` chooses `reps()` $=$ ceil$(1250 \times \alpha/2)/(\alpha/2) = 32/0.025 = 1280$. If an integer alternative cannot be found, the original `reps()` specification will be used. For details, see *Methods and formulas* below.

Because the WCB distribution is a step function, there is a range of values for each CI bound. For example, the lower-bound interval of values for the linear combination `tenure` − `ttl_exp` is $-0.191586 < a_{\text{lb}} \leq -0.191553$. The reported bounds are the largest value of each interval. We obtain -0.191586 by running `wildbootstrap` specifying `level(#)` with $\# = (0.95 + 2/1280) \times 100$. We would likely report the CIs using three digits of precision, for instance, $[-0.192, 0.150]$, so this interval is negligible but significant in a numerical root search where a convergence tolerance for r_l might be 10^{-8}.

The estimate of `tenure` − `ttl_exp` is reported to be -0.0391 with a 95% CI of $[-0.192, 0.150]$. We therefore fail to reject the hypothesis that `tenure` = `ttl_exp`. Similarly, the estimate of `1.collgrad` − `1.union` is 1.36 with a CI of $[-0.188, 3.91]$, and we conclude the difference of 1 is feasible.

◁

▷ Example 5: Fixed-effects linear regression with panels

Our final example demonstrates the use of `wildbootstrap` with estimator `xtreg`. Continuing with the data in the previous example, we use `industry` as the variable defining the panels.

We specify normal error weights, `errorweight(normal)`, thereby reducing the chance of identical draws for the error weights. Draws from the Rademacher distribution for 12 clusters have $2^{12} = 4096$ combinations.

We compute p-values and CIs on a subset of the model estimates: work tenure, work experience, and their interaction. If we replay the `xtreg` coefficient table, we will see that the fitted model includes `tenure`, `ttl_exp`, `c.tenure#c.ttl_exp`, `1.collgrad`, and `1.union`.

Finally, we use the symmetric p-value, option `ptype(symmetric)`, instead of the default equal-tailed, `ptype(equal)`. The table identifies the symmetric p-value with the header `P>|t|`.

The command and the results are as follows:

```
. xtset industry

Panel variable: industry (unbalanced)

. wildbootstrap xtreg wage c.tenure##c.ttl_exp i.collgrad i.union,
> rseed(12345) coef(ten ttl ten#ttl) errorweight(normal)
> ptype(symmetric)

Panel variable: industry (unbalanced)

Performing 1,000 replications for p-value for constraint
  tenure = 0 ...
Computing confidence interval for tenure
  Lower bound: .........10.........20...... done (26)
  Upper bound: .........10.........20. done (21)

Performing 1,000 replications for p-value for constraint
  ttl_exp = 0 ...
Computing confidence interval for ttl_exp
  Lower bound: .........10.........20...... done (26)
  Upper bound: .........10......... done (18)

Performing 1,000 replications for p-value for constraint
  c.tenure#c.ttl_exp = 0 ...
Computing confidence interval for c.tenure#c.ttl_exp
  Lower bound: .........10.........20...... done (26)
  Upper bound: .........10.........20... done (23)

Wild cluster bootstrap                          Number of obs      = 1,855
Fixed-effects linear regression                 Number of clusters =    12
                                                Cluster size:
Cluster variable: industry                                 min =     2
Error weight: Normal                                       avg = 154.6
                                                           max =   717
```

wage	Estimate	t	P>|t|	[95% conf.	interval]
constraints					
tenure = 0	.2026682	2.95	0.024	.0555676	.3753468
ttl_exp = 0	.2716375	11.59	0.000	.2227054	.3372929
c.tenure#c.ttl_exp = 0	-.0104125	-2.55	0.046	-.0201336	-.0003384

The fixed-effects model is the only `xtreg` model allowed with `wildbootstrap`, so specifying the `fe` option is not required.

Also note that the `cluster()` option is not specified. When the estimator `xtreg` is specified, the default cluster variable is the panel variable that is `xtset`. If the specified cluster variable is different

from the panel variable, then the levels of the panel variable must be nested within the cluster variable levels.

◁

Stored results

`wildbootstrap` stores the following in `e()`:

Scalars

`e(N_clust)`	number of clusters
`e(N_wbreps)`	number of bootstrap repetitions
`e(wb_block)`	bootstrap block size
`e(n_wbcns)`	number of bootstrap constraint restrictions
`e(min_c)`	smallest cluster size
`e(max_c)`	largest cluster size
`e(avg_c)`	average cluster size

Macros

`e(cmdline)`	command as typed
`e(cmd0)`	`wildbootstrap`
`e(wb_stat)`	WCB statistic, `t`
`e(wb_weight)`	WCB weights
`e(wb_ptype)`	WCB p-value criterion
`e(wb_level)`	WCB CI level
`e(wb_cistop)`	WCB CI interval type
`e(wb_rseed)`	random-number state
`e(wb_cns`i`)`	WCB constraint i, where i=1,...,`e(n_wbcns)`
`e(clustvar)`	name of cluster variable

Matrices

`e(wboot)`	WCB table
`e(wb_pci)`	WCB CI coverage
`e(wb_Cns)`	WCB constraint matrix

`wildbootstrap` will also carry forward most of the results already in `e()` from *command*.

In addition to the above, the following is stored in `r()`:

Matrices

`r(table)`	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

Introduction
CIs for linear combinations of coefficients
Constructing a CI inverting the hypothesis test

Introduction

We will focus our discussion of the WCB on linear regression, which easily extends to the other fixed-effects estimators. A linear regression model with clustered errors and G clusters can be written as

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_G \end{pmatrix} = \mathbf{X}\beta + \epsilon = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_G \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_G \end{pmatrix} \tag{1}$$

where for each cluster g, N_g is the number of observations, y_g is an $N_g \times 1$ vector of outcomes, X_g is an $N_g \times k$ matrix of covariates, and ϵ_g is an $N_g \times 1$ vector of errors. The parameter of interest is the $k \times 1$ vector of coefficients β. Error terms are assumed uncorrelated between observations of different clusters but possibly correlated between observations within the same cluster.

Without loss of generality, we will first focus on testing the null hypothesis H_0: $\beta_k = 0$ using the WCB algorithm. Let $\widehat{\beta}$ denote the ordinary least-squares estimator for β. We compute the t statistic for the kth coefficient as

$$t_k = \frac{\widehat{\beta}_k}{\sqrt{\widehat{\mathbf{V}}_{k,k}}}$$

where $\widehat{\mathbf{V}}_{k,k}$ is the CRVE for $\widehat{\beta}_k$. This is the kth diagonal element of the matrix

$$\widehat{\mathbf{V}} = \frac{G(N-1)}{(G-1)(N-k)} (\mathbf{X}\mathbf{X}')^{-1} \left(\sum_{g=1}^{G} \mathbf{X}_g' \widehat{\epsilon}_g \widehat{\epsilon}_g' \mathbf{X}_g \right) (\mathbf{X}\mathbf{X}')^{-1} \tag{2}$$

where $\widehat{\epsilon}_g$ is the $N_g \times 1$ vector of ordinary least-squares residuals for cluster g and $N = \sum_{g=1}^{G} N_g$ is the total number of observations.

The WCB algorithm proceeds as follows:

1. Refit model (1) subject to the restriction $\beta_k = 0$. Let $\widetilde{\beta}$ denote the restricted estimates and $\widetilde{\epsilon}$ denote the restricted residuals.
2. For each individual bootstrap replication b (out of a total of B replications):
   - 2a. Generate random variable ν_g^b for each cluster g according to the distribution specified in the `errorweight()` option.
   - 2b. For each cluster g and each observation in the cluster $i = 1, 2, \ldots, N_g$, generate a new bootstrap-dependent variable y_{ig}^b using the data-generating process:

     $$y_{ig}^b = X_{ig}\widetilde{\beta} + \widetilde{\epsilon}_{ig}\nu_g^b$$

   - 2c. Fit model (1) using the bootstrap variable y_{ig}^b as regressand. Calculate the t statistic for $\beta_k = 0$,

     $$t_k^b = \frac{\widehat{\beta}_k^b}{\sqrt{\widehat{\mathbf{V}}_{k,k}^b}}$$

     where $\widehat{\beta}_k^b$ and $\widehat{\mathbf{V}}_{k,k}^b$ are the ordinary least-squares coefficient and the CRVE for that coefficient in the bootstrap replication, respectively. In this case, the CRVE is the kth diagonal element of matrix (2) obtained when using the residuals from bootstrap replication b.

3. The p-values for the one-sided alternative hypotheses $H_1: \beta_k > 0$ and $H_2: \beta_k < 0$ are given by

$$p_1 = \frac{1}{B}\sum_{b=1}^{B} I\left(t_k^b > t_k\right)$$

$$p_2 = \frac{1}{B}\sum_{b=1}^{B} I\left(t_k^b < t_k\right)$$

For the alternative hypothesis $H_3: \beta_k \neq 0$, the p-value assuming that the distribution of the t statistic is symmetric around 0 is given by

$$p_S = \frac{1}{B}\sum_{b=1}^{B} I\left(|t_k^b| > |t_k|\right)$$

If the assumption of symmetry is not appropriate, then the p-value is given by the equal-tailed p-value $p_e = 2\min(p_1, p_2)$. See Djogbenou, MacKinnon, and Nielsen (2019).

To increase computational speed, at the expense of computer memory usage, the `wildbootstrap` command uses the matrix algebra of Roodman et al. (2019). By doing this, the computational complexity (run time) of the WCB algorithm is reduced from the order $O(NB)$ to the order $O(GB)$. The WCB uses a $G \times B$ matrix of random variables. By organizing these variables in column-major order, `wildbootstrap` reduces overall memory usage with minimal extra computation by breaking up the number of bootstrap replicates into blocks (the `blocksize(#)` option).

CIs for linear combinations of coefficients

By inverting hypotheses tests, we can apply the WCB algorithm to find CIs for any linear combination of coefficients. Suppose we wanted to compute a CI for the linear combination of parameters $\mathbf{R}\beta$, where $\mathbf{R}$ is a $1 \times k$ vector. In this case, the associated null hypothesis is $H_0: \mathbf{R}\beta = r$, where r is an arbitrary scalar. We can test this null hypothesis with the WCB algorithm described in the previous section by using the bootstrap t statistics

$$t^b(r) = \frac{\mathbf{R}\widehat{\beta}^b - r}{\sqrt{\mathbf{R}\widehat{\mathbf{V}}^b\mathbf{R}'}}$$

for $b = 1, \ldots, B$. For more on restricted regression with linear constraints on multiple coefficients, see [P] **makecns**.

The associated one-sided alternative hypotheses are now $H_1: \mathbf{R}\beta > r$ and $H_2: \mathbf{R}\beta < r$. As before, their respective bootstrap p-values are given by

$$p_1(r) = \frac{1}{B}\sum_{b=1}^{B} I\left\{t^b(r) > t(r)\right\}$$

$$p_2(r) = \frac{1}{B}\sum_{b=1}^{B} I\left\{t^b(r) < t(r)\right\}$$

where $t(r)$ is the t statistic from the original sample:

$$t(r) = \frac{\mathbf{R}\widehat{\beta} - r}{\sqrt{\mathbf{R}\widehat{\mathbf{V}}\mathbf{R}'}}$$

The associated two-sided alternative hypothesis is H_3: $\mathbf{R}\beta \neq r$, and its bootstrap p-value under the assumption of symmetry is given by

$$p_S(r) = \frac{1}{B}\sum_{b=1}^{B} I\left\{\left|t^b(r)\right| > |t(r)|\right\}$$

When using the equal-tailed criterion (the `ptype(equal)` option), the $100(1-\alpha)\%$ CI for $\mathbf{R}\beta$ is given by $[r_l, r_u]$, where r_l and r_u satisfy $p_1(r_l) = p_2(r_u) = \alpha/2$. On the other hand, when using the symmetric criterion (the `ptype(symmetric)` option), r_l and r_u satisfy $p_S(r_l) = p_S(r_u) = \alpha$. The parameter α can be specified with the `level()` option, that is, $\alpha = (100 - \texttt{level()})/100$.

The bootstrap p-values are step functions on r, and therefore a range of values for r_l and for r_u will solve the p-value conditions above. Because distribution functions are right continuous, `wildbootstrap` chooses the rightmost point in each range of solutions.

For the equal-tailed CIs, the starting values for the search of solutions to the p-value conditions are chosen as follows. First, the bootstrap t statistics are sorted $t^{(1)} \leq t^{(2)} \leq \ldots < t^{(B)}$. Second, we define $b_l = \text{ceil}\{B(1-\alpha/2)\}$ so that $t^{(b_l)}$ is smaller than $B(\alpha/2)$ of the t statistics. Similarly, we define $b_u = \text{floor}\{B(\alpha/2)\}$ so that $t^{(b_u)}$ is larger than $B(\alpha/2)$ of the t statistics. Third and finally, the initial lower and upper bounds of the CI are given by

$$r_l = \mathbf{R}\widehat{\beta} - t^{(b_l)}\sqrt{\mathbf{R}\widehat{\mathbf{V}}\mathbf{R}'}$$

$$r_u = \mathbf{R}\widehat{\beta} - t^{(b_u)}\sqrt{\mathbf{R}\widehat{\mathbf{V}}\mathbf{R}'}$$

where $\widehat{\beta}$ and $\widehat{\mathbf{V}}$ are the unrestricted estimator for β and the unrestricted CRVE in the original sample.

Constructing a CI inverting the hypothesis test

For a linear combination of coefficients $\sum R_j\beta_j$, we can construct a CI by inverting the hypothesis test H_0: $\mathbf{R}\beta = r$, where $\mathbf{R}$ is a row vector that includes the coefficients R_j and r is an arbitrary scalar. To do this, `wildbootstrap` searches for the CI lower bound r_l such that $\Pr(\mathbf{R}\beta \leq r_l) = \alpha/2$ (equal-tailed criterion, the `ptype(equal)` option). The WCB distribution of $\mathbf{R}\beta$ is a step function with a step size of $1/\texttt{reps(\#)}$. When $\texttt{reps(\#)} \times \alpha/2$ is an integer, call it S, then we search for the r_l that produces the ordered bootstrapped t statistics

$$t^{(b)} = \frac{\mathbf{R}\widehat{\boldsymbol{\beta}}^{(b)} - r_l}{\sqrt{\mathbf{R}'\mathbf{V}^{(b)}\mathbf{R}}}$$

with the property $\sum_{b=1}^{\text{reps()}} I(t^{(b)} > t) = S$ and therefore $\Pr(\mathbf{R}\beta \leq r_l) = S/\texttt{reps(\#)} = \alpha/2$.

Acknowledgments

We thank David Roodman of Open Philanthropy for his insights on the wild bootstrap computation.

We also acknowledge previous and ongoing contributions to this area from the Stata community. In particular, we acknowledge David Roodman, James G. MacKinnon, Morten Ørregaard Nielsen, and Matthew D. Webb for their command `boottest` and Mitchell Petersen, Douglas Miller, and Judson Caskey for their command `cgmwildboot`.

References

Bell, R. M., and D. F. McCaffrey. 2002. Bias reduction in standard errors for linear regression with multi-stage samples. *Survey Methodology* 28: 169–181.

Cameron, A. C., J. B. Gelbach, and D. L. Miller. 2008. Bootstrap-based improvements for inference with clustered errors. *Review of Economics and Statistics* 90: 414–427. https://doi.org/10.1162/rest.90.3.414.

Cameron, A. C., and D. L. Miller. 2015. A practitioner's guide to cluster-robust inference. *Journal of Human Resources* 50: 317–372. https://doi.org/10.3368/jhr.50.2.317.

Djogbenou, A. A., J. G. MacKinnon, and M. Ø. Nielsen. 2019. Asymptotic theory and wild bootstrap inference with clustered errors. *Journal of Econometrics* 212: 393–412. https://doi.org/10.1016/j.jeconom.2019.04.035.

MacKinnon, J. G. 2019. How cluster-robust inference is changing applied econometrics. *Canadian Journal of Economics* 52: 851–881. https://doi.org/10.1111/caje.12388.

MacKinnon, J. G., M. Ø. Nielsen, and M. D. Webb. 2023. Cluster-robust inference: A guide to empirical practice. *Journal of Econometrics* 232: 272–299. https://doi.org/10.1016/j.jeconom.2022.04.001.

MacKinnon, J. G., and M. D. Webb. 2017. Wild bootstrap inference for wildly different cluster sizes. *Journal of Applied Econometrics* 32: 233–254. https://doi.org/10.1002/jae.2508.

——. 2018. The wild bootstrap for few (treated) clusters. *Econometrics Journal* 21: 114–135. https://doi.org/10.1111/ectj.12107.

Roodman, D., J. G. MacKinnon, M. Ø. Nielsen, and M. D. Webb. 2019. Fast and wild: Bootstrap inference in Stata using boottest. *Stata Journal* 19: 4–60.

Wu, C. F. J. 1986. Jackknife, bootstrap and other resampling methods in regression analysis. *Annals of Statistics* 14: 1261–1350 (including discussions and rejoinder). https://doi.org/10.1214/aos/1176350142.

Also see

[R] **areg** — Linear regression with a large dummy-variable set

[R] **regress** — Linear regression

[XT] **xtreg** — Fixed-, between-, and random-effects and population-averaged linear models

[U] 20 Estimation and postestimation commands