Package ‘multiwayvcov’

February 20, 2015

Encoding UTF-8
Type Package
Title Multi-way Standard Error Clustering
Version 1.2.1
Date 2014-10-19
Author Nathaniel Graham and Mahmood Arai and Björn Hagströmer
Maintainer Nathaniel Graham <npgraham1@gmail.com>
Depends R (>= 3.0.0)
Imports sandwich, boot, compiler, parallel
Suggests lmtest
URL http://sites.google.com/site(npgraham1/research/code
LazyData no
Description Exports two functions implementing
multi-way clustering using the method suggested by Cameron, Gelbach, &
Miller (2011) and cluster (or block)
bootstrapping for estimating variance-covariance matrices. Normal one and
two-way clustering matches the results of other common statistical
packages. Missing values are handled transparently and rudimentary
parallelization support is provided.
License BSD_2_clause + file LICENSE
ByteCompile yes
NeedsCompilation no
Repository CRAN
Date/Publication 2014-10-20 10:30:20

R topics documented:

   cluster.boot .......................................................... 2
   cluster.vcov .......................................................... 4
   petersen ................................................................. 7

Index 8
cluster.boot  

**Bootstrapped multi-way standard error clustering**

**Description**

Return a bootstrapped multi-way cluster-robust variance-covariance matrix.

**Usage**

```r
cluster.boot(model, cluster, parallel = FALSE, use_white = NULL,
force_posdef = FALSE, R = 300, boot_type = "xy",
wild_type = "rademacher", debug = FALSE)
```

**Arguments**

- `model`  
  The estimated model, usually an `lm` or `glm` class object.

- `cluster`  
  A vector, matrix, or `data.frame` of cluster variables, where each column is a separate variable. If the vector `1:nrow(data)` is used, the function effectively produces a regular heteroskedasticity-robust matrix.

- `parallel`  
  Scalar or list. If a list, use the list as a list of connected processing cores/clusters. Scalar values of `true` and "snow" (which are equivalent) ask `boot` to handle parallelization, as does "multicore". See the parallel and boot package.

- `use_white`  
  Logical or null. See description below.

- `force_posdef`  
  Logical. Force the eigenvalues of the variance-covariance matrix to be positive.

- `R`  
  Integer. The number of bootstrap replicates; passed directly to `boot`.

- `boot_type`  
  "xy", "residual", or "wild". See details.

- `wild_type`  
  "rademacher", "mammen", or "norm". See details.

- `debug`  
  Logical. Print internal values useful for debugging to the console.

**Details**

This function implements cluster bootstrapping (also known as the block bootstrap) for variance-covariance matrices, following Cameron, Gelbach, & Miller (CGM) (2008). Usage is generally similar to the `cluster.vcov` function in this package, but this function does not support degrees of freedom corrections or leverage adjustments.

In the terminology that CGM (2008) use, this function implements pairs, residual, or wild cluster bootstrap-se.

A pairs (or xy) cluster bootstrap can be obtained by setting `boot_type = "xy"`, which resamples the entire regression data set (both X and y). Setting `boot_type = "residual"` will obtain a residual cluster bootstrap, which resamples only the residuals (in this case, we resample the blocks/clusters rather than the individual observations’ residuals). To get a wild cluster bootstrap set `boot_type = "wild"`, which does not resample anything, but instead reforms the dependent variable by multiplying the residual by a randomly drawn value and adding the result to the fitted value. The default method is the pairs/xy bootstrap.
There are three built-in distributions to draw multipliers from for wild bootstraps: the Rademacher (wild_type = "rademacher", the default), which draws from [-1, 1], each with P = 0.5, Mammen’s suggested distribution (wild_type = "mammen", see Mammen, 1993), and the standard normal/Gaussian distribution (wild_type = "norm"). The default is the Rademacher distribution, following CGM (2008). Alternatively, you can set the function to draw multipliers from by assigning wild_type to a function that takes no arguments and returns a single real value.

Multi-way clustering is handled as described by Petersen (2009) and generalized according to Cameron, Gelbach, & Miller (2011). This means that cluster.boot estimates a set of variance-covariance matrices for the variables separately and then sums them (subtracting some matrices and adding others). The method described by CGM (2011) estimates a set of variance-covariance matrices for the residuals (sometimes referred to as the meat of the sandwich estimator) and sums them appropriately. Whether you sum the meat matrices and then compute the model’s variance-covariance matrix or you compute a series of model matrices and sum those is mathematically irrelevant, but may lead to (very) minor numerical differences.

Unlike the cluster.vcov function, this function does not depend upon the estfun function from the sandwich package, although it does make use of the vcovHC function for computing White HC0 variance-covariance matrices.

Parallelization (if used) is handled by the boot package. Be sure to set options(boot.ncpus = N) where N is the number of CPU cores you want the boot function to use.

Value

a k x k variance-covariance matrix of type matrix

Author(s)

Nathaniel Graham <npgraham1@gmail.com>

References


Examples

```r
# Not run:
library(lmtest)
data(petersen)
m1 <- lm(y ~ x, data = petersen)

# Cluster by firm
boot_firm <- cluster.boot(m1, petersen$firmid)
```
# Cluster by year
boot_year <- cluster.boot(m1, petersen$year)
coeftest(m1, boot_year)

# Double cluster by firm and year
boot_both <- cluster.boot(m1, cbind(petersen$firmid, petersen$year))
coeftest(m1, boot_both)

# Cluster by firm with wild bootstrap and custom wild distribution
boot_firm2 <- cluster.boot(m1, petersen$firmid, boot_type = "wild",
                           wild_type = function() sample(c(-1, 1), 1))
coeftest(m1, boot_firm)

# Go multicore using the parallel package
require(parallel)
cl <- makeCluster(4)
options(boot.ncpus = 4)
boot_both <- cluster.boot(m1, cbind(petersen$firmid, petersen$year), parallel = cl)
stopCluster(cl)
coeftest(m1, boot_both)

# Go multicore using the parallel package, let boot handle the parallelization
require(parallel)
options(boot.ncpus = 8)
boot_both <- cluster.boot(m1, cbind(petersen$firmid, petersen$year), parallel = TRUE)
coeftest(m1, boot_both)

## End(Not run)

---

crystal.vcov

**Multi-way standard error clustering**

### Description

Return a multi-way cluster-robust variance-covariance matrix

### Usage

```r
cluster.vcov(model, cluster, parallel = FALSE, use_white = NULL,
             df_correction = TRUE, leverage = FALSE, force_posdef = FALSE,
             debug = FALSE)
```

### Arguments

- **model**: The estimated model, usually an `lm` or `glm` class object
- **cluster**: A vector, matrix, or data.frame of cluster variables, where each column is a separate variable. If the vector `1:nrow(data)` is used, the function effectively produces a regular heteroskedasticity-robust matrix.
**parallel**  Scalar or list. If a list, use the list as a list of connected processing cores/clusters. A scalar indicates no parallelization. See the parallel package.

**use_white**  Logical or NULL. See description below.

**df_correction**  Logical or vector. TRUE computes degrees of freedom corrections, FALSE uses no corrections. A vector of length $2^D - 1$ will directly set the degrees of freedom corrections.

**leverage**  Integer. EXPERIMENTAL Uses MacKinnon-White HC3-style leverage adjustments. Known to work in the non-clustering case, e.g., it reproduces HC3 if df_correction = FALSE. Set to 3 for HC3-style and 2 for HC2-style leverage adjustments.

**force_posdef**  Logical. Force the eigenvalues of the variance-covariance matrix to be positive.

**debug**  Logical. Print internal values useful for debugging to the console.

**Details**

This function implements multi-way clustering using the method suggested by Cameron, Gelbach, & Miller (2011), which involves clustering on $2^D - 1$ dimensional combinations, e.g., if we’re cluster on firm and year, then we compute for firm, year, and firm-year. Variance-covariance matrices with an odd number of cluster variables are added, and those with an even number are subtracted.

The cluster variable(s) are specified by passing the entire variable(s) to cluster (cbind()-ed as necessary). The cluster variables should be of the same number of rows as the original data set; observations omitted or excluded in the model estimation will be handled accordingly.

Ma (2014) suggests using the White (1980) variance-covariance matrix as the final, subtracted matrix when the union of the clustering dimensions U results in a single observation per group in U; e.g., if clustering by firm and year, there is only one observation per firm-year, we subtract the White (1980) HC0 variance-covariance from the sum of the firm and year vcov matrices. This is detected automatically (if use_white = NULL), but you can force this one way or the other by setting use_white = TRUE or FALSE.

Some authors suggest avoiding degrees of freedom corrections with multi-way clustering. By default, the function uses corrections identical to Petersen (2009) corrections. Passing a numerical vector to df_correction (of length $2^D - 1$) will override the default, and setting df_correction = FALSE will use no correction.

Cameron, Gelbach, & Miller (2011) further suggest a method for forcing the variance-covariance matrix to be positive semidefinite by correcting the eigenvalues of the matrix. To use this method, set force_posdef = TRUE. Do not use this method unless absolutely necessary! The eigen/spectral decomposition used is not ideal numerically, and may introduce small errors or deviations. If force_posdef = TRUE, the correction is applied regardless of whether it’s necessary.

The defaults deliberately match the Stata default output for one-way and Mitchell Petersen’s two-way Stata code results. To match the SAS default output (obtained using the class & repeated subject statements, see Arellano (1987)) simply turn off the degrees of freedom correction.

Parallelization is available via the parallel package by passing the “cluster” list (usually called c1) to the parallel argument.

**Value**

a $k \times k$ variance-covariance matrix of type ‘matrix’
Author(s)
Nathaniel Graham <npgraham1@gmail.com>

References


Ma, Mark (Shuai), Are We Really Doing What We Think We Are Doing? A Note on Finite-Sample Estimates of Two-Way Cluster-Robust Standard Errors (April 9, 2014).


Examples

```r
library(lmtest)
data(petersen)
m1 <- lm(y ~ x, data = petersen)

# Cluster by firm
vcov_firm <- cluster.vcov(m1, petersen$firmid)
coeftest(m1, vcov_firm)

# Cluster by year
vcov_year <- cluster.vcov(m1, petersen$year)
coeftest(m1, vcov_year)

# Double cluster by firm and year
vcov_both <- cluster.vcov(m1, cbind(petersen$firmid, petersen$year))
coeftest(m1, vcov_both)

# Replicate Mahmood Arai's double cluster by firm and year
vcov_both <- cluster.vcov(m1, cbind(petersen$firmid, petersen$year), use_white = FALSE)
coeftest(m1, vcov_both)

# For comparison, produce White HC0 VCOV the hard way
vcov_hc0 <- cluster.vcov(m1, 1:nrow(petersen), df_correction = FALSE)
coeftest(m1, vcov_hc0)

# Produce White HC1 VCOV the hard way
vcov_hc1 <- cluster.vcov(m1, 1:nrow(petersen), df_correction = TRUE)
coeftest(m1, vcov_hc1)

# Produce White HC2 VCOV the hard way
```
Simulation of clustering with firm and time effects.

Description

A dataset containing the 500 simulated firms over 10 years. Originally created by Mitchell Petersen in conjunction with Petersen (2009) and made available at http://www.kellogg.northwestern.edu/faculty/petersen/htm/papers/se/test_data.txt. See the references for simulation process. The variables are as follows:

Format

A data frame with 5000 rows and 4 variables

Details

- firmid. Firm identifier.
- year. Year identifier.
- x. Independent (right-hand side) variable.
- y. Dependent (left-hand side) variable.

References


Mitchell Petersen’s description of the simulation process: http://www.kellogg.northwestern.edu/faculty/petersen/htm/papers/se/se_programming.htm
Index

*Topic block
  cluster.boot, 2

*Topic bootstrap
  cluster.boot, 2

*Topic boot
  cluster.boot, 2

*Topic clustering
  cluster.boot, 2
  cluster.vcov, 4

*Topic datasets
  petersen, 7

*Topic errors
  cluster.boot, 2
  cluster.vcov, 4

*Topic multi-way
  cluster.boot, 2
  cluster.vcov, 4

*Topic robust
  cluster.boot, 2
  cluster.vcov, 4

*Topic standard
  cluster.boot, 2
  cluster.vcov, 4

  petersen, 7