

**xporegress** — Cross-fit partialing-out lasso linear regression

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">Reference</a>	<a href="#">Also see</a>		

## Description

`xporegress` fits a lasso linear regression model and reports coefficients along with standard errors, test statistics, and confidence intervals for specified covariates of interest. The cross-fit partialing-out method is used to estimate effects for these variables and to select from potential control variables to be included in the model.

## Quick start

Estimate a coefficient for `d1` in a linear regression of `y` on `d1`, and include `x1` to `x100` as potential control variables to be selected by lassos

```
xporegress y d1, controls(x1-x100)
```

Same as above, and estimate coefficients for the levels of categorical `d2`

```
xporegress y d1 i.d2, controls(x1-x100)
```

Same as above, but use 20 folds instead of 10 for cross-fitting

```
xporegress y d1 i.d2, controls(x1-x100) xfolds(20)
```

Same as above, but repeat the cross-fitting procedure 15 times, and average the results

```
xporegress y d1 i.d2, controls(x1-x100) xfolds(20) resample(15)
```

Use cross-validation (CV) instead of a plugin iterative formula to select the optimal  $\lambda^*$  in each lasso

```
xporegress y d1 i.d2, controls(x1-x100) selection(cv)
```

Same as above, and set a random-number seed for reproducibility

```
xporegress y d1 i.d2, controls(x1-x100) selection(cv) rseed(28)
```

Specify CV for the lasso for `y` only, with the stopping rule criterion turned off

```
xporegress y d1 i.d2, controls(x1-x100) lasso(y, selection(cv), stop(0))
```

Same as above, but apply the option to the lassos for `y`, `d1`, and `i.d2`

```
xporegress y d1 i.d2, controls(x1-x100) lasso(*, selection(cv), stop(0))
```

## Menu

Statistics > Lasso > Lasso inferential models > Continuous outcomes > Cross-fit partialing-out model

## Syntax

```

xporegress depvar varsofinterest [if] [in],
           controls([(alwaysvars)] othervars) [options]

```

*varsofinterest* are variables for which coefficients and their standard errors are estimated.

<i>options</i>	Description
<b>Model</b>	
* <u>controls</u> ( <i>[(alwaysvars)] othervars</i> )	<i>alwaysvars</i> and <i>othervars</i> make up the set of control variables; <i>alwaysvars</i> are always included; lassos choose whether to include or exclude <i>othervars</i>
<u>selection</u> (plugin)	use a plugin iterative formula to select an optimal value of the lasso penalty parameter $\lambda^*$ for each lasso; the default
<u>selection</u> (cv)	use CV to select an optimal value of the lasso penalty parameter $\lambda^*$ for each lasso
<u>selection</u> (adaptive)	use adaptive lasso to select an optimal value of the lasso penalty parameter $\lambda^*$ for each lasso
<u>selection</u> (bic)	use BIC to select an optimal value of the lasso penalty parameter $\lambda^*$ for each lasso
<u>sqr</u> lasso	use square-root lassos
<u>xfolds</u> (#)	use # folds for cross-fitting
<u>resample</u> (#)	repeat sample splitting # times and average results
<u>technique</u> (dm11   dm12)	use either double machine learning 1 (dm11) or double machine learning 2 (dm12) estimation technique; dm12 is the default
<u>semi</u>	use semipartialing-out lasso regression estimator
<u>missing</u> ok	after fitting lassos, ignore missing values in any <i>othervars</i> not selected, and include these observations in the final model
<b>SE/Robust</b>	
<u>vce</u> ( <i>vcetype</i> )	<i>vcetype</i> may be <u>robust</u> (the default) or <u>cluster</u> <i>clustvar</i>
<b>Reporting</b>	
<u>level</u> (#)	set confidence level; default is level(95)
<u>display_options</u>	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
<b>Optimization</b>	
[ <u>no</u> ]log	display or suppress an iteration log
<u>verbose</u>	display a verbose iteration log
<u>rseed</u> (#)	set random-number seed
<b>Advanced</b>	
<u>lasso</u> ( <i>varlist</i> , <i>lasso_options</i> )	specify options for the lassos for variables in <i>varlist</i> ; may be repeated
<u>sqr</u> lasso( <i>varlist</i> , <i>lasso_options</i> )	specify options for square-root lassos for variables in <i>varlist</i> ; may be repeated

<code>reestimate</code>	refit the model after using <code>lassoselect</code> to select a different $\lambda^*$
<code>noheader</code>	do not display the header on the coefficient table
<code>coeflegend</code>	display legend instead of statistics

\*`controls()` is required.

`varsofinterest`, `alwaysvars`, and `othervars` may contain factor variables. Base levels of factor variables cannot be set for `alwaysvars` and `othervars`. See [U] 11.4.3 Factor variables.

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

`reestimate`, `noheader`, and `coeflegend` do not appear in the dialog box.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Options

### Model

`controls`(`[(alwaysvars)]` *othervars*) specifies the set of control variables, which control for omitted variables. Control variables are also known as confounding variables. `xprogress` fits lassos for *devar* and each of the *varsofinterest*. *alwaysvars* are variables that are always to be included in these lassos. *alwaysvars* are optional. *othervars* are variables that each lasso will choose to include or exclude. That is, each lasso will select a subset of *othervars* and other lassos will potentially select different subsets of *othervars*. `controls()` is required.

`selection`(`plugin|cv|adaptive|bic`) specifies the selection method for choosing an optimal value of the lasso penalty parameter  $\lambda^*$  for each lasso or square-root lasso estimation. Separate lassos are estimated for *devar* and each variable in *varsofinterest*. Specifying `selection()` changes the selection method for all of these lassos. You can specify different selection methods for different lassos using the option `lasso()` or `sqrtlasso()`. When `lasso()` or `sqrtlasso()` is used to specify a different selection method for the lassos of some variables, they override the global setting made using `selection()` for the specified variables.

`selection(plugin)` is the default. It selects  $\lambda^*$  based on a “plugin” iterative formula dependent on the data. See [LASSO] lasso options.

`selection(cv)` selects the  $\lambda^*$  that gives the minimum of the CV function. See [LASSO] lasso options.

`selection(adaptive)` selects  $\lambda^*$  using the adaptive lasso selection method. It cannot be specified when `sqrtlasso` is specified. See [LASSO] lasso options.

`selection(bic)` selects the  $\lambda^*$  that gives the minimum of the BIC function. See [LASSO] lasso options.

`sqrtlasso` specifies that square-root lassos be done rather than regular lassos. The option `lasso()` can be used with `sqrtlasso` to specify that regular lasso be done for some variables, overriding the global `sqrtlasso` setting for these variables. See [LASSO] lasso options.

`xfolds(#)` specifies the number of folds for cross-fitting. The default is `xfolds(10)`.

`resample`(`[(#)]`) specifies that sample splitting be repeated and results averaged. This reduces the effects of the randomness of sample splitting on the estimated coefficients. Not specifying `resample` or `resample(#)` is equivalent to specifying `resample(1)`. In other words, by default no resampling is done. Specifying `resample` alone is equivalent to specifying `resample(10)`. That is, sample splitting is repeated 10 times. For each sample split, lassos are computed. So when this option is not specified, lassos are repeated `xfolds(#)` times. But when `resample(#)` is specified, lassos are repeated `xfolds(#)`  $\times$  `resample(#)` times. Thus, while we recommend using `resample` to get final results, note that it can be an extremely time-consuming procedure.

`technique(dm11|dm12)` specifies which cross-fitting technique is used, either double machine learning 1 (`dm11`) or double machine learning 2 (`dm12`). For both techniques, the initial estimation steps are the same. The sample is split into  $K = \text{xfolds}(\#)$  folds. Then, coefficients on the controls are estimated using only the observations not in the  $k$ th fold, for  $k = 1, 2, \dots, K$ . Moment conditions for the coefficients on the *varsofinterest* are formed using the observations in fold  $k$ . The default technique, `dm12`, solves the moment conditions jointly across all the observations. The optional technique, `dm11`, solves the moment conditions in each fold  $k$  to produce  $K$  different estimates, which are then averaged to form a single vector of estimates. See [Methods and formulas](#).

`semi` specifies that the semipartialing-out lasso regression estimator be used instead of the fully partialing-out lasso estimator, which is the default. See [Methods and formulas](#).

`missingok` specifies that, after fitting lassos, the estimation sample be redefined based on only the nonmissing observations of variables in the final model. In all cases, any observation with missing values for *deivar*, *varsofinterest*, *alwaysvars*, and *othervars* is omitted from the estimation sample for the lassos. By default, the same sample is used for calculation of the coefficients of the *varsofinterest* and their standard errors.

When `missingok` is specified, the initial estimation sample is the same as the default, but the sample used for the calculation of the coefficients of the *varsofinterest* can be larger. Now observations with missing values for any *othervars* not selected will be added to the estimation sample (provided there are no missing values for any of the variables in the final model).

`missingok` may produce more efficient estimates when data are missing completely at random. It does, however, have the consequence that estimation samples can change when selected variables differ in models fit using different selection methods. That is, when *othervars* contain missing values, the estimation sample for a model fit using the default `selection(plugin)` will likely differ from the estimation sample for a model fit using, for example, `selection(cv)`.

#### SE/Robust

`vce(vctype)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification (`robust`) and that allow for intragroup correlation (`cluster clustvar`); see [\[R\] vce\\_option](#).

When `vce(cluster clustvar)` is specified, all lassos also account for clustering. For each lasso, this affects how the log-likelihood function is computed and how the sample is split in cross-validation; see [Methods and formulas](#) in [\[LASSO\] lasso](#). Specifying `vce(cluster clustvar)` may lead to different selected controls and therefore to different point estimates for your variable of interest when compared to the estimation that ignores clustering.

#### Reporting

`level(#)`; see [\[R\] Estimation options](#).

`display_options`: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(style)`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [\[R\] Estimation options](#).

#### Optimization

`[no]` `log` displays or suppresses a log showing the progress of the estimation. By default, one-line messages indicating when each lasso estimation begins are shown. Specify `verbose` to see a more detailed log.

`verbose` displays a verbose log showing the iterations of each lasso estimation. This option is useful when doing `selection(cv)` or `selection(adaptive)`. It allows you to monitor the progress

of the lasso estimations for these selection methods, which can be time consuming when there are many *othervars* specified in `controls()`.

`rseed(#)` sets the random-number seed. This option can be used to reproduce results. `rseed(#)` is equivalent to typing `set seed #` prior to running `xporegress`. Random numbers are used to produce split samples for cross-fitting. So for all `selection()` options, if you want to reproduce your results, you must either use this option or use `set seed`. See [R] [set seed](#).

Advanced

`lasso(varlist, lasso_options)` lets you set different options for different lassos, or advanced options for all lassos. You specify a *varlist* followed by the options you want to apply to the lassos for these variables. *varlist* consists of one or more variables from *devar* or *varsofinterest*. `_all` or `*` may be used to specify *devar* and all *varsofinterest*. This option is repeatable as long as different variables are given in each specification. *lasso\_options* are `selection(...)`, `grid(...)`, `stop(#)`, `tolerance(#)`, `dtolerance(#)`, and `cvtolerance(#)`. When `lasso(varlist, selection(...))` is specified, it overrides any global `selection()` option for the variables in *varlist*. It also overrides the global `sqrtlasso` option for these variables. See [LASSO] [lasso options](#).

`sqrtlasso(varlist, lasso_options)` works like the option `lasso()`, except square-root lassos for the variables in *varlist* are done rather than regular lassos. *varlist* consists of one or more variables from *devar* or *varsofinterest*. This option is repeatable as long as different variables are given in each specification. *lasso\_options* are `selection(...)`, `grid(...)`, `stop(#)`, `tolerance(#)`, `dtolerance(#)`, and `cvtolerance(#)`. When `sqrtlasso(varlist, selection(...))` is specified, it overrides any global `selection()` option for the variables in *varlist*. See [LASSO] [lasso options](#).

The following options are available with `xporegress` but are not shown in the dialog box:

`reestimate` is an advanced option that refits the `xporegress` model based on changes made to the underlying lassos using `lassoselect`. After running `xporegress`, you can select a different  $\lambda^*$  for one or more of the lassos estimated by `xporegress`. After selecting  $\lambda^*$ , you type `xporegress, reestimate` to refit the `xporegress` model based on the newly selected  $\lambda^*$ 's.

`reestimate` may be combined only with reporting options.

`noheader` prevents the coefficient table header from being displayed.

`coeflegend`; see [R] [Estimation options](#).

## Remarks and examples

[stata.com](http://www.stata.com)

`xporegress` performs cross-fit partialing-out lasso linear regression. This command estimates coefficients, standard errors, and confidence intervals and performs tests for variables of interest while using lassos to select from among potential control variables.

The linear regression model is

$$E[y|\mathbf{d}, \mathbf{x}] = \mathbf{d}\alpha' + \mathbf{x}\beta'$$

where  $\mathbf{d}$  are the variables for which we wish to make inferences and  $\mathbf{x}$  are the potential control variables from which the lassos select. `xporegress` reports estimated coefficients for  $\alpha$ . However, cross-fit partialing-out does not provide estimates of the coefficients on the control variables ( $\beta$ ) or their standard errors. No estimation results can be reported for  $\beta$ .

For an introduction to the cross-fit partialing-out lasso method for inference, as well as the double-selection and partialing-out methods, see [\[LASSO\] Lasso inference intro](#).

Examples that demonstrate how to use `xporegress` and the other lasso inference commands are presented in [\[LASSO\] Inference examples](#). In particular, we recommend reading [1 Overview](#) for an introduction to the examples and to the `v1` command, which provides tools for working with the large lists of variables that are often included when using lasso methods. See [2 Fitting and interpreting inferential models](#) for examples of fitting inferential lasso linear models and comparisons of the different methods available in Stata.

If you are interested in digging deeper into the lassos that are used to select controls, see [5 Exploring inferential model lassos](#) in [\[LASSO\] Inference examples](#).

## Stored results

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

### Scalars

<code>e(N)</code>	number of observations
<code>e(N_clust)</code>	number of clusters
<code>e(k_varsofinterest)</code>	number of variables of interest
<code>e(k_controls)</code>	number of potential control variables
<code>e(k_controls_sel)</code>	number of selected control variables
<code>e(df)</code>	degrees of freedom for test of variables of interest
<code>e(chi2)</code>	$\chi^2$
<code>e(p)</code>	$p$ -value for test of variables of interest
<code>e(n_xfolds)</code>	number of folds for cross-fitting
<code>e(n_resample)</code>	number of resamples
<code>e(rank)</code>	rank of $e(V)$

### Macros

<code>e(cmd)</code>	<code>xporegress</code>
<code>e(cmdline)</code>	command as typed
<code>e(depvar)</code>	name of dependent variable
<code>e(lasso_depvars)</code>	names of dependent variables for all lassos
<code>e(varsofinterest)</code>	variables of interest
<code>e(controls)</code>	potential control variables
<code>e(controls_sel)</code>	selected control variables
<code>e(model)</code>	<code>linear</code>
<code>e(title)</code>	title in estimation output
<code>e(clustvar)</code>	name of cluster variable
<code>e(chi2type)</code>	Wald; type of $\chi^2$ test
<code>e(vce)</code>	<code>vcetype</code> specified in <code>vce()</code>
<code>e(vcetype)</code>	title used to label Std. err.
<code>e(rngstate)</code>	random-number state used
<code>e(properties)</code>	<code>b V</code>
<code>e(predict)</code>	program used to implement <code>predict</code>
<code>e(select_cmd)</code>	program used to implement <code>lassoselect</code>
<code>e(marginsnotok)</code>	predictions disallowed by <code>margins</code>
<code>e(asbalanced)</code>	factor variables <code>fvset</code> as <code>asbalanced</code>
<code>e(asobserved)</code>	factor variables <code>fvset</code> as <code>asobserved</code>

### Matrices

<code>e(b)</code>	coefficient vector
<code>e(V)</code>	variance–covariance matrix of the estimators

### Functions

<code>e(sample)</code>	marks estimation sample
------------------------	-------------------------

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

Matrices	
<code>r(table)</code>	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

`xporegress` implements cross-fit partialing-out lasso linear regression as described in Chernozhukov et al. (2018), where they derived two versions of cross-fitting that are known as double machine learning 1 (DML1) and double machine learning 2 (DML2). DML2 is the default method and corresponds with option `technique(dml2)`. Specify option `technique(dml1)` to get DML1 instead.

Methods DML1 and DML2 have a similar structure. Each does the following.

1. Partitions the sample into  $K$  folds.
2. Uses the postlasso estimates computed using the observations not in a specific fold to fill in the moment conditions for the observations in that fold.

DML1 solves the moment conditions using the observations in each fold to produce  $K$  different estimates and then averages these  $K$  estimates to produce the final estimate for the coefficients of interest. DML2 uses all the observations to solve the moment conditions to produce a single final estimate for the coefficients of interest.

`xporegress` implements two methods for the partialing-out lasso regression. We call the default method partialing-out lasso regression (POLR). We call the optional method, obtained by specifying option `semi`, a semipartialing-out lasso regression (SPOLR). See *Methods and formulas* in [LASSO] **po**regress for a brief literature review of POLR and SPOLR.

The regression model is

$$\mathbf{E}[y|\mathbf{d}, \mathbf{x}] = \mathbf{d}\boldsymbol{\alpha}' + \beta_0 + \mathbf{x}\boldsymbol{\beta}'$$

where  $\mathbf{d}$  contains the  $J$  covariates of interest and  $\mathbf{x}$  contains the  $p$  controls. The number of covariates in  $\mathbf{d}$  must be small and fixed. The number of controls in  $\mathbf{x}$  can be large and, in theory, can grow with the sample size; however, the number of nonzero elements in  $\boldsymbol{\beta}$  must not be too large, which is to say that the model must be sparse.

### Cross-fit POLR algorithm

1. Randomly partition the sample into  $K$  subsamples called folds.
2. Define  $I_k$  to be the observations in fold  $k$ , and define  $IC_k$  to be the sample observations not in fold  $k$ .
3. For each  $k = 1, \dots, K$ , fill in the observations of  $i \in I_k$  for the  $J$  moment conditions that identify  $\boldsymbol{\alpha}$ . These moment conditions use out-of-sample estimates of the high-dimensional components estimated using the observations  $i \in IC_k$ .

- a. Using the observations  $i \in IC_k$ , perform a linear lasso of  $y$  on  $\mathbf{x}$  to select covariates  $\tilde{\mathbf{x}}_{k,y}$ .

This lasso can choose the lasso penalty parameter ( $\lambda^*$ ) using one of the plugin estimators for a linear lasso, adaptive lasso, or CV. The heteroskedastic plugin estimator for the linear lasso is the default.

- b. Using the observations  $i \in IC_k$ , fit a linear regression of  $y$  on  $\tilde{\mathbf{x}}_{k,y}$ , and let  $\hat{\boldsymbol{\delta}}_k$  be the estimated coefficients on  $\tilde{\mathbf{x}}_{k,y}$ .

c. For the observations  $i \in I_k$ , fill in the partial outcome.

$$\tilde{y}_i = y_i - \tilde{\mathbf{x}}_{k,y,i} \hat{\boldsymbol{\delta}}_k'$$

d. Using the observations  $i \in IC_k$ , for each  $j = 1, \dots, J$ , perform a linear lasso of  $d_j$  on  $\mathbf{x}$  to select covariates  $\tilde{\mathbf{x}}_{k,j}$ .

Each of these lassos can choose the lasso penalty parameter ( $\lambda_j^*$ ) using one of the plugin estimators for a linear lasso, adaptive lasso, or CV. The heteroskedastic plugin estimator for the linear lasso is the default.

e. Using the observations  $i \in IC_k$ , for  $j = 1, \dots, J$ , fit a linear regression of  $d_j$  on  $\tilde{\mathbf{x}}_{k,j}$ , and denote the coefficient estimates by  $\hat{\boldsymbol{\gamma}}_{k,j}$ .

f. For each observation  $i \in I_k$ , and for  $j = 1, \dots, J$ , fill in the instrument

$$z_{j,i} = d_{j,i} - \tilde{\mathbf{x}}_{k,j,i} \hat{\boldsymbol{\gamma}}_{k,j}'$$

g. For each observation  $i \in I_k$ , collect the instruments into a vector  $\mathbf{z}_i = (z_{1,i}, z_{2,i}, \dots, z_{J,i})$ .

4. Compute the point estimates.

For DML2, compute  $\hat{\boldsymbol{\alpha}}$  by solving the following sample-moment equations.

$$\frac{1}{n} \sum_{i=1}^n \mathbf{z}_i' (\tilde{y}_i - \mathbf{z}_i \boldsymbol{\alpha}') = \mathbf{0}$$

For DML1,  $\hat{\boldsymbol{\alpha}}$  is given by

$$\hat{\boldsymbol{\alpha}} = \frac{1}{K} \sum_{k=1}^K \hat{\boldsymbol{\alpha}}_k$$

where  $\hat{\boldsymbol{\alpha}}_k$  is computed by solving the sample-moment equations

$$\frac{1}{n_k} \sum_{i \in I_k} \mathbf{z}_i' (\tilde{y}_i - \mathbf{z}_i \boldsymbol{\alpha}'_k) = \mathbf{0}$$

and  $n_k$  is the number of observations in  $I_k$ .

5. The VCE is estimated by

$$\widehat{\mathbf{Var}}(\hat{\boldsymbol{\alpha}}) = \frac{1}{n} \hat{\mathbf{J}}_0^{-1} \hat{\boldsymbol{\Psi}} \left( \hat{\mathbf{J}}_0^{-1} \right)'$$



where

$$\begin{aligned}\widehat{\Psi} &= \frac{1}{K} \sum_{k=1}^K \widehat{\Psi}_k \\ \widehat{\Psi}_k &= \frac{1}{n_k} \sum_{i \in I_k} \widehat{\psi}_i \widehat{\psi}_i' \\ \widehat{\psi}_i &= \mathbf{z}_i' (\widetilde{y}_i - \mathbf{z}_i \widehat{\alpha}') \\ \widehat{\mathbf{J}}_0 &= \frac{1}{K} \sum_{k=1}^K \left( \frac{1}{n_k} \sum_{i \in I_k} \widehat{\psi}_i^a \right)\end{aligned}$$

and

$$\widehat{\psi}_i^a = \mathbf{z}_i' \mathbf{z}_i$$

### Cross-fit SPOLR algorithm

1. Randomly partition the sample into  $K$  subsamples called folds.
2. Define  $I_k$  to be the observations in fold  $k$ , and define  $IC'_k$  to be the sample observations not in fold  $k$ .
3. For each  $k = 1, \dots, K$ , fill in the observations of  $i \in I_k$  for the  $J$  moment conditions that identify  $\alpha$ . These moment conditions use out-of-sample estimates of the high-dimensional components estimated using the observations  $i \in IC'_k$ .

- a. Using the observations  $i \in IC'_k$ , perform a linear lasso of  $y$  on  $\mathbf{d}$  and  $\mathbf{x}$  to select covariates  $\widetilde{\mathbf{x}}_{k,y}$ .

This lasso can choose the lasso penalty parameter ( $\lambda^*$ ) using one of the plugin estimators for a linear lasso, adaptive lasso, or CV. The heteroskedastic plugin estimator for the linear lasso is the default.

- b. Using the observations  $i \in IC'_k$ , fit a linear regression of  $y$  on  $\mathbf{d}$  and  $\widetilde{\mathbf{x}}_{k,y}$ , and let  $\widehat{\delta}_k$  be the estimated coefficients on  $\widetilde{\mathbf{x}}_{k,y}$ .
- c. For the observations  $i \in I_k$ , fill in the partial outcome.

$$\widetilde{y}_i = y_i - \widetilde{\mathbf{x}}_{k,y,i} \widehat{\delta}_k'$$

- d. Using the observations  $i \in IC'_k$ , for each  $j = 1, \dots, J$ , perform a linear lasso of  $d_j$  on  $\mathbf{x}$  to select covariates  $\widetilde{\mathbf{x}}_{k,j}$ .

Each of these lassos can choose the lasso penalty parameter ( $\lambda_j^*$ ) using one of the plugin estimators for a linear lasso, adaptive lasso, or CV. The heteroskedastic plugin estimator for the linear lasso is the default.

- e. Using the observations  $i \in IC'_k$ , for  $j = 1, \dots, J$ , fit a linear regression of  $d_j$  on  $\widetilde{\mathbf{x}}_{k,j}$ , and denote the coefficient estimates by  $\widehat{\gamma}_{k,j}$ .

f. For each observation  $i \in I_k$ , and for  $j = 1, \dots, J$ , fill in the instrument

$$z_{j,i} = d_{j,i} - \tilde{\mathbf{x}}_{k,j,i} \hat{\gamma}_{k,j}$$

g. For each observation  $i \in I_k$ , collect the instruments into a vector  $\mathbf{z}_i = (z_{1,i}, z_{2,i}, \dots, z_{J,i})$ .

4. Compute the point estimates.

For DML2, compute  $\hat{\boldsymbol{\alpha}}$  by solving the following sample-moment equations.

$$\frac{1}{n} \sum_{i=1}^n (\tilde{y}_i - \mathbf{d}_i \boldsymbol{\alpha}') \mathbf{z}'_i = \mathbf{0}$$

For DML1,  $\hat{\boldsymbol{\alpha}}$  is given by

$$\hat{\boldsymbol{\alpha}} = \frac{1}{K} \sum_{k=1}^K \hat{\boldsymbol{\alpha}}_k$$

where  $\hat{\boldsymbol{\alpha}}_k$  is computed by solving the sample-moment equations

$$\frac{1}{n_k} \sum_{i \in I_k} (\tilde{y}_i - \mathbf{d}_i \hat{\boldsymbol{\alpha}}'_k) \mathbf{z}'_i = \mathbf{0}$$

and  $n_k$  is the number of observations in  $I_k$ .

5. The VCE is estimated by

$$\widehat{\text{Var}}(\hat{\boldsymbol{\alpha}}) = \frac{1}{n} \hat{\mathbf{J}}_0^{-1} \hat{\boldsymbol{\Psi}} \left( \hat{\mathbf{J}}_0^{-1} \right)'$$

where

$$\hat{\boldsymbol{\Psi}} = \frac{1}{K} \sum_{k=1}^K \hat{\boldsymbol{\Psi}}_k$$

$$\hat{\boldsymbol{\Psi}}_k = \frac{1}{n_k} \sum_{i \in I_k} \hat{\boldsymbol{\psi}}_i \hat{\boldsymbol{\psi}}'_i$$

$$\hat{\boldsymbol{\psi}}_i = (\tilde{y}_i - \mathbf{d}_i \hat{\boldsymbol{\alpha}}') \mathbf{z}'_i$$

$$\hat{\mathbf{J}}_0 = \frac{1}{K} \sum_{k=1}^K \left( \frac{1}{n_k} \sum_{i \in I_k} \hat{\boldsymbol{\psi}}_i^a \right)$$

and

$$\boldsymbol{\psi}_i^a = \mathbf{d}_i \mathbf{z}'_i$$

See *Methods and formulas* in [LASSO] **lasso** for details on how the lassos in steps 3a and 3d of both algorithms choose their penalty parameters ( $\lambda^*$ ).

### Resampling the partitions

The  $K$  folds are chosen once by default. Specify option `resample(#)` to have the  $K$  folds randomly selected  $\#$  times. This resampling removes the dependence of the estimator on any specifically selected folds, at the cost of more computer time.

Let  $S$  be the specified number of resamples.

1. For each random partition  $s = 1, \dots, S$ , use a cross-fit estimator to obtain the DM1 or the DM2 point estimates  $\hat{\alpha}_s$  and the estimated VCE  $\widehat{\text{Var}}(\hat{\alpha}_s)$ .
2. The mean resampling-corrected point estimates are

$$\tilde{\alpha} = \frac{1}{S} \sum_{s=1}^S \hat{\alpha}_s$$

3. The mean resampling-corrected estimate of the VCE is

$$\widetilde{\text{Var}}(\tilde{\alpha}) = \frac{1}{S} \sum_{s=1}^S \left\{ \widehat{\text{Var}}(\hat{\alpha}_s) + (\hat{\alpha}_s - \tilde{\alpha})(\hat{\alpha}_s - \tilde{\alpha})' \right\}$$

## Reference

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. B. Hansen, W. K. Newey, and J. M. Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *Econometrics Journal* 21: C1–C68. <https://doi.org/10.1111/ectj.12097>.

## Also see

[LASSO] **lasso inference postestimation** — Postestimation tools for lasso inferential models

[LASSO] **dsregress** — Double-selection lasso linear regression

[LASSO] **poregress** — Partialing-out lasso linear regression

[R] **regress** — Linear regression

[U] **20 Estimation and postestimation commands**

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow and NetCourseNow are trademarks of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2023 StataCorp LLC, College Station, TX, USA. All rights reserved.



For suggested citations, see the FAQ on [citing Stata documentation](#).