Title

stata.com

xpologit — Cross-fit partialing-out lasso logistic regression

Description

`xpologit` fits a lasso logistic regression model and reports odds ratios 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

Report an odds ratio from a logistic regression of `y` on `d1`, and include `x1` to `x100` as potential control variables to be selected by lassos

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

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

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

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

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

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

```
xpologit 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 λ^* in each lasso

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

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

```
xpologit 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

```
xpologit 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`

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

Menu

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

Syntax

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

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

options	Description
Model	
* `controls([(alwaysvars)] othervars)`	*alwaysvars* and *othervars* make up the set of control variables; *alwaysvars* are always included; lassos choose whether to include or exclude *othervars*
`selection(plugin)`	use a plugin iterative formula to select an optimal value of the lasso penalty parameter λ^* for each lasso; the default
`selection(cv)`	use CV to select an optimal value of the lasso penalty parameter λ^* for each lasso
`selection(adaptive)`	use adaptive lasso to select an optimal value of the lasso penalty parameter λ^* for each lasso
`selection(bic)`	use BIC to select an optimal value of the lasso penalty parameter λ^* for each lasso
`sqrtlasso`	use square-root lassos
`xfolds(#)`	use # folds for cross-fitting
`resample[(#)]`	repeat sample splitting # times and average results
`technique(dml1 | dml2)`	use either double machine learning 1 (`dml1`) or double machine learning 2 (`dml2`) estimation technique; `dml2` is the default
`missingok`	after fitting lassos, ignore missing values in any *othervars* not selected, and include these observations in the final model
`offset(`*varname*`)`	include *varname* in the lasso and model for *depvar* with its coefficient constrained to be 1
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `robust` (the default) or `cluster` *clustvar*
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`or`	report odds ratios; the default
`coef`	report estimated coefficients
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Optimization	
`[no]log`	display or suppress an iteration log
`verbose`	display a verbose iteration log
`rseed(#)`	set random-number seed

Advanced

`lasso(`*varlist*`,` *lasso_options*`)`	specify options for the lassos for variables in *varlist*; may be repeated
`sqrtlasso(`*varlist*`,` *lasso_options*`)`	specify options for square-root lassos for variables in *varlist*; may be repeated
`reestimate`	refit the model after using `lassoselect` to select a different λ^*
`noheader`	do not display the header on the coefficient table
`coeflegend`	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. `xpologit` fits lassos for *depvar* 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 λ^* for each lasso or square-root lasso estimation. Separate lassos are estimated for *depvar* 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 λ^* based on a "plugin" iterative formula dependent on the data. See [LASSO] **lasso options**.

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

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

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

`sqrtlasso` specifies that square-root lassos be done rather than regular lassos for the *varsofinterest*. This option does not apply to *depvar*. Square-root lassos are linear models, and the lasso for *depvar* is always a logit lasso. 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(dml1|dml2)` specifies which cross-fitting technique is used, either double machine learning 1 (`dml1`) or double machine learning 2 (`dml2`). For both techniques, the initial estimation steps are the same. The sample is split into $K =$ `xfolds(#)` folds. Then, coefficients on the controls are estimated using only the observations not in the kth fold, for $k = 1, 2, \ldots, K$. Moment conditions for the coefficients on the *varsofinterest* are formed using the observations in fold k. The default technique, `dml2`, solves the moment conditions jointly across all the observations. The optional technique, `dml1`, 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*.

`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 *depvar*, *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)`.

`offset(`*varname*`)` specifies that *varname* be included in the lasso and model for *depvar* with its coefficient constrained to be 1.

SE/Robust

`vce(`*vcetype*`)` 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**.

or reports the estimated coefficients transformed to odds ratios, that is, e^{α}. Standard errors and confidence intervals are similarly transformed. or is the default.

coef reports the estimated coefficients α rather than the odds ratios (e^{α}). This option affects how results are displayed, not how they are estimated. coef may be specified at estimation or when replaying previously estimated results.

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 xpologit. 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 *depvar* or *varsofinterest*. _all or * may be used to specify *depvar* 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 *varsofinterest*. Square-root lassos are linear models, and this option cannot be used with *depvar*. 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 xpologit but are not shown in the dialog box:

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

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

The logistic regression model is

$$\Pr(y = 1|\mathbf{d}, \mathbf{x}) = \frac{\exp(\mathbf{d}\boldsymbol{\alpha}' + \mathbf{x}\boldsymbol{\beta}')}{1 + \exp(\mathbf{d}\boldsymbol{\alpha}' + \mathbf{x}\boldsymbol{\beta}')}$$

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. `xpologit` estimates the $\boldsymbol{\alpha}$ coefficients and reports the corresponding odds ratios, e^{α}. However, cross-fit partialing-out does not provide estimates of the coefficients on the control variables ($\boldsymbol{\beta}$) or their standard errors. No estimation results can be reported for $\boldsymbol{\beta}$.

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 `xpologit` 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 `vl` 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 comparisons of the different methods of fitting inferential models that are available in Stata. Everything we say there about methods of selection is applicable to both linear and nonlinear models. See *3 Fitting logit inferential models to binary outcomes. What is different?* for examples and discussion specific to logistic regression models. The primary difference from linear models involves interpreting the results.

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

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

Scalars

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

Macros

`e(cmd)`	`xpologit`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(lasso_depvars)`	names of dependent variables for all lassos
`e(varsofinterest)`	variables of interest
`e(controls)`	potential control variables
`e(controls_sel)`	selected control variables
`e(model)`	`logit`

e(title)	title in estimation output
e(offset)	linear offset variable
e(clustvar)	name of cluster variable
e(chi2type)	Wald; type of χ^2 test
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(rngstate)	random-number state used
e(properties)	b V
e(predict)	program used to implement predict
e(select_cmd)	program used to implement lassoselect
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(V)	variance–covariance matrix of the estimators

Functions

e(sample)	marks estimation sample

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

xpologit implements cross-fit partialing-out lasso logit regression (XPOLLR) 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.

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. See *Methods and formulas* in [LASSO] **xporegress** for details about resampling.

The regression model is

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

where $G(a) = \exp(a)/\{1 + \exp(a)\}$, $\mathbf{d}$ 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.

XPOLLR 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 $k = 1, \ldots, K$, 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 logit lasso of y on $\mathbf{d}$ and $\mathbf{x}$ to select controls $\widetilde{\mathbf{x}}_{k,y}$.

      This logit lasso can choose the lasso penalty parameter (λ_k^*) using the plugin estimator, adaptive lasso, or CV. The plugin value is the default.
   b. Using the observations $i \in IC_k$, fit a logit regression of y on $\mathbf{d}$ and $\widetilde{\mathbf{x}}_{k,y}$, let $\widetilde{\boldsymbol{\alpha}}_k$ be the estimated coefficients on $\mathbf{d}$, and let $\widetilde{\boldsymbol{\delta}}_k$ be the estimated coefficients on $\widetilde{\mathbf{x}}_{k,y}$.
   c. For the observations $i \in I_k$, fill in the prediction for the high-dimensional component using the out-of-sample estimate $\widetilde{\boldsymbol{\delta}}_k$.

$$\widetilde{s}_i = \widetilde{\mathbf{x}}_{k,y,i}\widetilde{\boldsymbol{\delta}}_k'$$

   d. Using the observations $i \in IC_k$, for $j = 1, \ldots, J$, perform a linear lasso of d_j on $\mathbf{x}$ using observation-level weights

$$w_i = G'(\mathbf{d}_i\widetilde{\boldsymbol{\alpha}}_k' + \widetilde{s}_i)$$

      where $G'()$ is the derivative of $G()$, and denote the selected controls by $\widetilde{\mathbf{x}}_{k,j}$.

      Each of these lassos can choose the lasso penalty parameter (λ_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 $j = 1, \ldots, J$, fit a linear regression of d_j on $\widetilde{\mathbf{x}}_{k,j}$, and denote the coefficient estimates by $\widehat{\boldsymbol{\gamma}}_{k,j}$.
   f. For each observation $i \in I_k$, and for $j = 1, \ldots, J$, fill in the instrument

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

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

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

$$\frac{1}{n}\sum_{i=1}^{n}\{y_i - G(\mathbf{d}_i\boldsymbol{\alpha}' + \widetilde{s}_i)\}\,\mathbf{z}_i' = \mathbf{0}$$

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

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

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

$$\frac{1}{n_k}\sum_{i\in I_k}\{y_i - G(\mathbf{d}_i\boldsymbol{\alpha}_k' + \widetilde{s}_i)\}\,\mathbf{z}_i' = \mathbf{0}$$

and n_k is the number of observations in I_k.

5. The VCE is estimated by

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

where

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

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

$$\widehat{\boldsymbol{\psi}}_i = \left\{y_i - G(\mathbf{d}_i\widehat{\boldsymbol{\alpha}}' + \widetilde{s}_i)\right\}\mathbf{z}_i'$$

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

and

$$\widehat{\boldsymbol{\psi}}_i^a = \frac{\partial\widehat{\boldsymbol{\psi}}_i}{\partial\widehat{\boldsymbol{\alpha}}}$$

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

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] **dslogit** — Double-selection lasso logistic regression

[LASSO] **pologit** — Partialing-out lasso logistic regression

[R] **logit** — Logistic regression, reporting coefficients

[R] **logistic** — Logistic regression, reporting odds ratios

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