lasso options — Lasso options for inferential models

Description Syntax Options Remarks and examples Reference Also see

Description

This entry describes the options that control the lassos, either individually or globally, in the `ds`, `po`, and `xpo` estimation commands.

For an introduction to lasso inferential models, see [LASSO] **Lasso inference intro**.

For examples of the `ds`, `po`, and `xpo` estimation commands and the use of these options, see [LASSO] **Inference examples**.

Syntax

lasso_inference_cmd ... [, ...*options*]

lasso_inference_cmd is one of `dslogit`, `dspoisson`, `dsregress`, `poivregress`, `pologit`, `popoisson`, `poregress`, `xpoivregress`, `xpologit`, `xpopoisson`, or `xporegress`.

options	Description
Model	
`selection(plugin)`	select λ^* using a plugin iterative formula for all lassos; the default
`selection(cv)`	select λ^* using cross-validation (CV) for all lassos
`selection(adaptive)`	select λ^* using adaptive lasso for all lassos
`selection(bic)`	select λ^* using Bayesian information criterion (BIC) for all lassos
`sqrtlasso`	fit square-root lassos instead of regular lassos
Advanced	
`lasso(`*varlist*`,` *lasso_options*`)`	specify options for lassos for variables in *varlist*
`sqrtlasso(`*varlist*`,` *lasso_options*`)`	specify options for square-root lassos for variables in *varlist*

lasso_options	Description
`selection(`*sel_method*`)`	selection method to select an optimal value of the lasso penalty parameter λ^* from the set of possible λ's
`grid(`$\#_g$ [`, ratio(#) min(#)`]`)`	specify the set of possible λ's using a logarithmic grid with $\#_g$ grid points
`stop(#)`	tolerance for stopping the iteration over the λ grid early
`cvtolerance(#)`	tolerance for identification of the CV function minimum
`bictolerance(#)`	tolerance for identification of the BIC function minimum
`tolerance(#)`	convergence tolerance for coefficients based on their values
`dtolerance(#)`	convergence tolerance for coefficients based on deviance

sel_method	Description
plugin[, *plugin_opts*]	select λ^* using a plugin iterative formula; the default
cv[, *cv_opts*]	select λ^* using CV
adaptive[, *adapt_opts cv_opts*]	select λ^* using an adaptive lasso; only available for `lasso()`
bic[, *bic_opts*]	select λ^* using BIC

plugin_opts	Description
heteroskedastic	assume model errors are heteroskedastic; the default
homoskedastic	assume model errors are homoskedastic

cv_opts	Description
folds(#)	use # folds for CV
alllambdas	fit models for all λ's in the grid or until the stop(#) tolerance is reached; by default, the CV function is calculated sequentially by λ, and estimation stops when a minimum is identified
serule	use the one-standard-error rule to select λ^*
stopok	when the CV function does not have an identified minimum and the stop(#) stopping criterion for λ was reached at λ_{stop}, set the selected λ^* to be λ_{stop}; the default
strict	do not select λ^* when the CV function does not have an identified minimum; this is a stricter alternative to the default stopok
gridminok	when the CV function does not have an identified minimum and the stop(#) stopping criterion for λ was not reached, set the selected λ^* to be the minimum of the λ grid, λ_{gmin}; this is a looser alternative to the default stopok and is rarely used

adapt_opts	Description
steps(#)	use # adaptive steps (counting the initial lasso as step 1)
unpenalized	use the unpenalized estimator to construct initial weights
ridge	use the ridge estimator to construct initial weights
power(#)	raise weights to the #th power

bic_opts	Description
alllambdas	fit models for all λ's in the grid or until the stop(#) tolerance is reached; by default, the BIC function is calculated sequentially by λ, and estimation stops when a minimum is identified
stopok	when the BIC function does not have an identified minimum and the stop(#) stopping criterion for λ was reached at λ_{stop}, set the selected λ^* to be λ_{stop}; the default
strict	do not select λ^* when the BIC function does not have an identified minimum; this is a stricter alternative to the default stopok
gridminok	when the BIC function does not have an identified minimum and the stop(#) stopping criterion for λ was not reached, set the selected λ^* to be the minimum of the λ grid, λ_{gmin}; this is a looser alternative to the default stopok and is rarely used
postselection	use postselection coefficients to compute BIC

Options

Model

selection(plugin | cv | adaptive | bic) is a global option that specifies that all lassos use the given selection method. It is the same as specifying lasso(*, selection(plugin | cv | adaptive | bic)). The default is selection(plugin). That is, not specifying this option implies a global selection(plugin) for all lassos. This global form of the option does not allow suboptions. To specify suboptions, use the lasso() or sqrtlasso() option described below.

sqrtlasso is a global option that specifies that all lassos be square-root lassos. It is the same as specifying sqrtlasso(*), except for logit and Poisson models. For logit and Poisson models, it is the same as sqrtlasso(*varsofinterest*), where *varsofinterest* are all the variables that have lassos excluding the dependent variable. This global form of the option does not allow suboptions. To specify suboptions, use the sqrtlasso() option described below.

Advanced

lasso(*varlist*, *lasso_options*) and sqrtlasso(*varlist*, *lasso_options*) let you set different options for different lassos and square-root lassos. These options also let you specify advanced options for all lassos and all square-root lassos. The lasso() and sqrtlasso() options override the global options selection(plugin | cv | adaptive) and sqrtlasso for the lassos for the specified variables. If lasso(*varlist*, *lasso_options*) or sqrtlasso(*varlist*, *lasso_options*) does not contain a selection() specification as part of *lasso_options*, then the global option for selection() is assumed.

lasso(*varlist*, *lasso_options*) specifies that the variables in *varlist* be fit using lasso with the selection method, set of possible λ's, and convergence criteria determined by *lasso_options*.

sqrtlasso(*varlist*, *lasso_options*) specifies that the variables in *varlist* be fit using square-root lasso with the selection method, set of possible λ's, and convergence criteria determined by *lasso_options*.

For lasso() and sqrtlasso(), *varlist* consists of one or more variables from *depvar*, the dependent variable, or *varsofinterest*, the variables of interest. To specify options for all lassos, you may use * or _all to specify *depvar* and all *varsofinterest*.

For models with endogeneity, namely, `poivregress` and `xpoivregress` models, lassos are done for *depvar*, the exogenous variables, *exovars*, and the endogenous variables, *endovars*. Any of these variables can be specified in the `lasso()` option. All of them can be specified using `*` or `_all`.

The `lasso()` and `sqrtlasso()` options are repeatable as long as different variables are given in each specification of `lasso()` and `sqrtlasso()`. The type of lasso for any *depvar* or *varsofinterest* (or *exovars* or *endovars*) not specified in any `lasso()` or `sqrtlasso()` option is determined by the global lasso options described above.

For all lasso inferential commands, linear lassos are done for each of the *varsofinterest* (or *exovars* and *endovars*). For linear models, linear lassos are also done for *depvar*. For logit models, however, logit lassos are done for *depvar*. For Poisson models, Poisson lassos are done for *depvar*. Square-root lassos are linear models, so `sqrtlasso(`*depvar*`, ...)` cannot be specified for the dependent variable in logit and Poisson models. For the same reason, `sqrtlasso(*, ...)` and `sqrtlasso(_all, ...)` cannot be specified for logit and Poisson models. For logit and Poisson models, you must specify `sqrtlasso(`*varsofinterest*`, ...)` to set options for square-root lassos and specify `lasso(`*depvar*`, ...)` to set options for the logit or Poisson lasso for *depvar*.

Suboptions for lasso( ) and sqrtlasso( )

`selection(plugin` [`, heteroskedastic homoskedastic`]`)` selects λ^* based on a "plugin" iterative formula dependent on the data. The plugin estimator calculates a value for λ^* that dominates the noise in the estimating equations, which ensures that the variables selected belong to the true model with high probability. See *Methods and formulas* in [LASSO] **lasso**.

`selection(plugin)` does not estimate coefficients for any other values of λ, so it does not require a λ grid, and none of the grid options apply. It is much faster than the other selection methods because estimation is done only for a single value of λ. It is an iterative procedure, however, and if the plugin is computing estimates for a small λ (which means many nonzero coefficients), the estimation can still be time consuming.

`heteroskedastic` assumes model errors are heteroskedastic. It is the default. Specifying `selection(plugin)` for linear lassos is equivalent to specifying `selection(plugin, heteroskedastic)`. This suboption can be specified only for linear lassos. Hence, this suboption cannot be specified for *depvar* for logit and Poisson models, where *depvar* is the dependent variable. For these models, specify `lasso(`*depvar*`, selection(plugin))` to have the logit or Poisson plugin formula used for the lasso for *depvar*. See *Methods and formulas* in [LASSO] **lasso**.

`homoskedastic` assumes model errors are homoskedastic. This suboption can be specified only for linear lassos. Hence, this suboption cannot be specified for *depvar* for logit and Poisson models, where *depvar* is the dependent variable.

`selection(cv` [`, folds(`#`) alllambdas serule stopok strict gridminok`]`)` selects λ^* to be the λ that gives the minimum of the CV function.

`folds(`#`)` specifies that CV with # folds be done. The default is `folds(10)`.

`alllambdas` specifies that models be fit for all λ's in the grid or until the `stop(`#`)` tolerance is reached. By default, models are calculated sequentially from largest to smallest λ, and the CV function is calculated after each model is fit. If a minimum of the CV function is found, the computation ends at that point without evaluating additional smaller λ's.

`alllambdas` computes models for these additional smaller λ's. Because computation time is greater for smaller λ, specifying `alllambdas` may increase computation time manyfold. Specifying `alllambdas` is typically done only when a full plot of the CV function is wanted for assurance that a true minimum has been found. Regardless of whether `alllambdas` is specified, the selected λ^* will be the same.

`serule` selects λ^* based on the "one-standard-error rule" recommended by Hastie, Tibshirani, and Wainwright (2015, 13–14) instead of the λ that minimizes the CV function. The one-standard-error rule selects the largest λ for which the CV function is within a standard error of the minimum of the CV function.

`stopok`, `strict`, and `gridminok` specify what to do when the CV function does not have an identified minimum. A minimum is identified at λ^* when the CV function at both larger and smaller adjacent λ is greater than it is at λ^*. When the CV function has an identified minimum, `stopok`, `strict`, and `gridminok` all do the same thing: the selected λ^* is the λ that gives the minimum.

In some cases, however, the CV function declines monotonically as λ gets smaller and never rises to identify a minimum. When the CV function does not have an identified minimum, `stopok` and `gridminok` make alternative selections for λ^*, and `strict` makes no selection. You may specify only one of `stopok`, `strict`, or `gridminok`; `stopok` is the default if you do not specify one. With each of these suboptions, estimation results are always left in place, and alternative λ^* can be selected and evaluated.

`stopok` specifies that when the CV function does not have an identified minimum and the `stop(#)` stopping tolerance for λ was reached, the selected λ^* is λ_{stop}, the λ that met the stopping criterion. λ_{stop} is the smallest λ for which coefficients are estimated, and it is assumed that λ_{stop} has a CV function value close to the true minimum. When no minimum is identified and the `stop(#)` criterion is not met, an error is issued.

`strict` requires the CV function to have an identified minimum. If it does not, an error is issued.

`gridminok` is a rarely used suboption that specifies that when the CV function has no identified minimum and the `stop(#)` stopping criterion was not met, λ_{gmin}, the minimum of the λ grid, is the selected λ^*.

The `gridminok` selection criterion is looser than the default `stopok`, which is looser than `strict`. With `strict`, only an identified minimum is selected. With `stopok`, either the identified minimum or λ_{stop} is selected. With `gridminok`, either the identified minimum or λ_{stop} or λ_{gmin} is selected, in this order.

`selection(adaptive` [`, steps(#) unpenalized ridge power(#)` *cv_options*]`)` can be specified only as a suboption for `lasso()`. It cannot be specified as a suboption for `sqrtlasso()`. It selects λ^* using the adaptive lasso selection method. It consists of multiple lassos with each lasso step using CV. Variables with zero coefficients are discarded after each successive lasso, and variables with nonzero coefficients are given penalty weights designed to drive small coefficient estimates to zero in the next step. Hence, the final model typically has fewer nonzero coefficients than a single lasso.

`selection(bic` [`,` *bic_opts*]`)` selects λ^* to be the λ that gives the minimum of the BIC function.

bic_opts are `alllambdas`, `stopok`, `strict`, `gridminok`, and `postselection`.

`alllambdas` specifies that models be fit for all λ's in the grid or until the `stop(#)` tolerance is reached. By default, models are calculated sequentially from largest to smallest λ, and the BIC function is calculated after each model is fit. If a minimum of the BIC function is found, the computation ends at that point without evaluating additional smaller λ's.

alllambdas computes models for these additional smaller λ's. Because computation time is greater for smaller λ, specifying alllambdas may increase computation time manyfold. Specifying alllambdas is typically done only when a full plot of the BIC function is wanted for assurance that a true minimum has been found. Regardless of whether alllambdas is specified, the selected λ^* will be the same.

stopok, strict, and gridminok specify what to do when the BIC function does not have an identified minimum. A minimum is identified at λ^* when the BIC function at both larger and smaller adjacent λ's is greater than it is at λ^*. When the BIC function has an identified minimum, these options all do the same thing: the selected λ^* is the λ that gives the minimum. In some cases, however, the BIC function declines monotonically as λ gets smaller and never rises to identify a minimum. When the BIC function does not have an identified minimum, stopok and gridminok make alternative selections for λ^*, and strict makes no selection. You may specify only one of stopok, strict, or gridminok; stopok is the default if you do not specify one. With each of these options, estimation results are always left in place, and alternative λ^* can be selected and evaluated.

stopok specifies that when the BIC function does not have an identified minimum and the stop(#) stopping tolerance for λ was reached, the selected λ^* is λ_{stop}, the λ that met the stopping criterion. λ_{stop} is the smallest λ for which coefficients are estimated, and it is assumed that λ_{stop} has a BIC function value close to the true minimum. When no minimum is identified and the stop(#) criterion is not met, an error is issued.

strict requires the BIC function to have an identified minimum, and if not, an error is issued.

gridminok is a rarely used option that specifies that when the BIC function has no identified minimum and the stop(#) stopping criterion was not met, then λ_{gmin}, the minimum of the λ grid, is the selected λ^*.

The gridminok selection criterion is looser than the default stopok, which is looser than strict. With strict, only an identified minimum is selected. With stopok, either the identified minimum or λ_{stop} is selected. With gridminok, either the identified minimum or λ_{stop} or λ_{gmin} is selected, in this order.

postselection specifies to use the postselection coefficients to compute the BIC function. By default, the penalized coefficients are used.

steps(#) specifies that adaptive lasso with # lassos be done. By default, # = 2. That is, two lassos are run. After the first lasso estimation, terms with nonzero coefficients β_i are given penalty weights equal to $1/|\beta_i|$, terms with zero coefficients are omitted, and a second lasso is estimated. Terms with small coefficients are given large weights, making it more likely that small coefficients become zero in the second lasso. Setting # > 2 can produce more parsimonious models. See *Methods and formulas* in [LASSO] **lasso**

unpenalized specifies that the adaptive lasso use the unpenalized estimator to construct the initial weights in the first lasso. unpenalized is useful when CV cannot find a minimum. unpenalized cannot be specified with ridge.

ridge specifies that the adaptive lasso use the ridge estimator to construct the initial weights in the first lasso. ridge cannot be specified with unpenalized.

power(#) specifies that the adaptive lasso raise the weights to the #th power. The default power is 1. The specified power must be in the interval $[\,0.25, 2\,]$.

cv_options are all the suboptions that can be specified for `selection(cv)`, namely, `folds(#)`, `alllambdas`, `serule`, `stopok`, `strict`, and `gridminok`. The suboptions `alllambdas`, `strict`, and `gridminok` apply only to the first lasso estimated. For second and subsequent lassos, `gridminok` is the default. When `ridge` is specified, `gridminok` is automatically used for the first lasso.

`grid(`$\#_g$ [`, ratio(#) min(#)`]`)` specifies the set of possible λ's using a logarithmic grid with $\#_g$ grid points.

$\#_g$ is the number of grid points for λ. The default is $\#_g = 100$. The grid is logarithmic with the ith grid point ($i = 1, \ldots, n = \#_g$) given by $\ln \lambda_i = [(i-1)/(n-1)] \ln r + \ln \lambda_{\text{gmax}}$, where $\lambda_{\text{gmax}} = \lambda_1$ is the maximum, $\lambda_{\text{gmin}} = \lambda_n =$ `min(#)` is the minimum, and $r = \lambda_{\text{gmin}}/\lambda_{\text{gmax}} =$ `ratio(#)` is the ratio of the minimum to the maximum.

`ratio(#)` specifies $\lambda_{\text{gmin}}/\lambda_{\text{gmax}}$. The maximum of the grid, λ_{gmax}, is set to the smallest λ for which all the coefficients in the lasso are estimated to be zero (except the coefficients of the *alwaysvars*). λ_{gmin} is then set based on `ratio(#)`. When $p < N$, where p is the total number of *othervars* and *alwaysvars* (not including the constant term) and N is the number of observations, the default value of `ratio(#)` is 1e−4. When $p \geq N$, the default is 1e−2.

`min(#)` sets λ_{gmin}. By default, λ_{gmin} is based on `ratio(#)` and λ_{gmax}, which is computed from the data.

`stop(#)` specifies a tolerance that is the stopping criterion for the λ iterations. The default is 1e−5. This suboption does not apply when the selection method is `selection(plugin)`. Estimation starts with the maximum grid value, λ_{gmax}, and iterates toward the minimum grid value, λ_{gmin}. When the relative difference in the deviance produced by two adjacent λ grid values is less than `stop(#)`, the iteration stops and no smaller λ's are evaluated. The value of λ that meets this tolerance is denoted by λ_{stop}. Typically, this stopping criterion is met before the iteration reaches λ_{gmin}.

Setting `stop(#)` to a larger value means that iterations are stopped earlier at a larger λ_{stop}. To produce coefficient estimates for all values of the λ grid, `stop(0)` can be specified. Note, however, that computations for small λ's can be extremely time consuming. In terms of time, when using `selection(cv)`, `selection(adaptive)`, or `selection(bic)`, the optimal value of `stop(#)` is the largest value that allows estimates for just enough λ's to be computed to identify the minimum of the CV or BIC function. When setting `stop(#)` to larger values, be aware of the consequences of the default λ^* selection procedure given by the default `stopok`. You may want to override the `stopok` behavior by using `strict`.

`cvtolerance(#)` is a rarely used option that changes the tolerance for identifying the minimum CV function. For linear models, a minimum is identified when the CV function rises above a nominal minimum for at least three smaller λ's with a relative difference in the CV function greater than #. For nonlinear models, at least five smaller λ's are required. The default is 1e−3. Setting # to a bigger value makes a stricter criterion for identifying a minimum and brings more assurance that a declared minimum is a true minimum, but it also means that models may need to be fit for additional smaller λ, which can be time consuming. See *Methods and formulas* for [LASSO] **lasso** for more information about this tolerance and the other tolerances.

`bictolerance(#)` is a rarely used option that changes the tolerance for identifying the minimum BIC function. A minimum is identified when the BIC function rises above a nominal minimum for at least two smaller λ's with a relative difference in the BIC function greater than #. The default is 1e−2. Setting # to a bigger value makes a stricter criterion for identifying a minimum and brings more

assurance that a declared minimum is a true minimum, but it also means that models may need to be fit for additional smaller λ, which can be time consuming. See *Methods and formulas* in [LASSO] **lasso** for more information about this tolerance and the other tolerances.

`tolerance(#)` is a rarely used option that specifies the convergence tolerance for the coefficients. Convergence is achieved when the relative change in each coefficient is less than this tolerance. The default is `tolerance(1e-7)`.

`dtolerance(#)` is a rarely used option that changes the convergence criterion for the coefficients. When `dtolerance(#)` is specified, the convergence criterion is based on the change in deviance instead of the change in the values of coefficient estimates. Convergence is declared when the relative change in the deviance is less than #. More-accurate coefficient estimates are typically achieved by not specifying this option and instead using the default `tolerance(1e-7)` criterion or specifying a smaller value for `tolerance(#)`.

Remarks and examples

All the options shown here may seem overwhelming. However, you will likely never need to use many of them.

You would typically use the global options to change the selection method for each of the lassos performed by one of the lasso inference commands. For example, you can specify `selection(cv)`, `selection(adaptive)`, or `selection(bic)` to change the selection method globally from the default `selection(plugin)`.

Sometimes, CV fails to identify a minimum of the CV function and so fails to select λ^*; thus, the inferential command fails. Lasso inference postestimation commands provide tools to see what happened. Then it may be possible to set options so that an acceptable λ^* is selected. Of course, in many cases, the issue is not with the computation but rather with model specification or simply not having enough data.

To understand the `selection(cv)`, `selection(adaptive)`, and `selection(bic)` selection methods and how to set options to control them, you should first become familiar with lasso for prediction and model selection.

Notice, however, that options for the `lasso` and `sqrtlasso` commands are specified slightly differently than they are when used as suboptions for lasso inference commands. For instance, with `lasso`, you might specify `selection(cv, folds(20))`. With `dsregress` or one of the other inference commands, you would specify `lasso(*, selection(cv, folds(20)))` to specify that CV with 20 folds be used to select λ^* for each lasso.

Read [LASSO] **lasso** and [LASSO] **lasso fitting** to learn about the lasso options in greater detail.

Reference

Hastie, T. J., R. J. Tibshirani, and M. Wainwright. 2015. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Boca Raton, FL: CRC Press. https://doi.org/10.1201/b18401.

Also see

[LASSO] **Lasso intro** — Introduction to lasso

[LASSO] **Lasso inference intro** — Introduction to inferential lasso models

[LASSO] **lasso** — Lasso for prediction and model selection

[LASSO] **lasso fitting** — The process (in a nutshell) of fitting lasso models