Title

coefpath — Plot path of coefficients after lasso

Description Quick start Menu Syntax
Options Remarks and examples Also see

Description

`coefpath` graphs the coefficient paths after any lasso fit using `selection(cv)`, `selection(adaptive)`, `selection(bic)`, or `selection(none)`. A line is drawn for each coefficient that traces its value over the searched values of the lasso penalty parameter λ or over the ℓ_1-norm of the fitted coefficients that result from lasso selection using those values of λ.

`coefpath` can be used after `lasso`, `elasticnet`, `sqrtlasso`, `telasso`, or any of the lasso inference commands.

Quick start

Graph the coefficient paths after `lasso`, `sqrtlasso`, or `elasticnet`

    coefpath

Graph the unstandardized coefficient paths

    coefpath, rawcoefs

Graph the coefficient paths after `elasticnet` for the $\alpha = 0.5$ lasso

    coefpath, alpha(.5)

As above, but graph the paths using a single linestyle, rather than line-specific linestyles

    coefpath, alpha(.5) mono

After any of the `ds` or `po` commands, graph the paths for the dependent variable `y`

    coefpath, for(y)

As above, but graph the paths as a function of $\ln\lambda$

    coefpath, for(y) xunits(lnlambda)

After an `xpo` command without `resample`, graph the paths for `x` in cross-fit fold 2

    coefpath, for(x) xfold(2)

After an `xpo` command with `resample`, graph the paths for `x` in cross-fit fold 2 for the first resample

    coefpath, for(x) xfold(2) resample(1)

After `telasso`, graph the paths for the outcome variable `y` at treatment level 1

    coefpath, for(y) tlevel(1)

Menu

Statistics > Postestimation

Syntax

After `lasso`, `sqrtlasso`, *and* `elasticnet`

`coefpath` [, *options*]

After `ds` *and* `po` *commands*

`coefpath, for(`*varspec*`)` [*options*]

After `xpo` *commands without* `resample`

`coefpath, for(`*varspec*`) xfold(`*#*`)` [*options*]

After `xpo` *commands with* `resample`

`coefpath, for(`*varspec*`) xfold(`*#*`) resample(`*#*`)` [*options*]

After `telasso` *for the outcome variable*

`coefpath, for(`*varspec*`) tlevel(`*#*`)` [*options*]

After `telasso` *for the treatment variable*

`coefpath, for(`*varspec*`)` [*options*]

After `telasso` *for the outcome variable with cross-fitting but without* `resample`

`coefpath, for(`*varspec*`) tlevel(`*#*`) xfold(`*#*`)` [*options*]

After `telasso` *for the treatment variable with cross-fitting but without* `resample`

`coefpath, for(`*varspec*`) xfold(`*#*`)` [*options*]

After `telasso` *for the outcome variable with cross-fitting and* `resample`

`coefpath, for(`*varspec*`) tlevel(`*#*`) xfold(`*#*`) resample(`*#*`)` [*options*]

After `telasso` *for the treatment variable with cross-fitting and* `resample`

`coefpath, for(`*varspec*`) xfold(`*#*`) resample(`*#*`)` [*options*]

varspec is *varname*, except after `poivregress` and `xpoivregress`, when it is either *varname* or `pred(`*varname*`)`.

options	Description
Main	
`xunits(`*x_unit_spec*`)`	x-axis units (scale); default is `xunits(l1norm)`
`minmax`	adds minimum and maximum values to the x axis
* `for(`*varspec*`)`	lasso for *varspec*; `telasso`, `ds`, `po`, and `xpo` commands only
* `xfold(`*#*`)`	lasso for the #th cross-fit fold; `xpo` commands and `telasso` with `xfolds` only
* `resample(`*#*`)`	lasso for the #th resample; `xpo` commands and `telasso` with `resample` only
* `tlevel(`*#*`)`	lasso for the outcome model with the treatment level #; `telasso` only
`alpha(`*#*`)`	graph coefficient paths for $\alpha = \#$; default is the selected value α^*; only allowed after `elasticnet`
`rawcoefs`	graph unstandardized coefficient paths
Reference line	
`rlopts(`*cline_options*`)`	affect rendition of reference line
`norefline`	suppress plotting reference line
Path	
`lineopts(`*cline_options*`)`	affect rendition of all coefficient paths; not allowed when there are 100 or more coefficients
`line#opts(`*cline_options*`)`	affect rendition of coefficient path #; not allowed when there are 100 or more coefficients
`mono`	graph coefficient paths using a single line; default is `mono` for 100 or more coefficients
`monoopts(`*cline_options*`)`	affect rendition of line used to graph coefficient paths when `mono` is specified
Data	
`data(`*filename* [`, replace`]`)`	save plot data to *filename*
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than `by()` documented in [G-3] ***twoway_options***

* `for(`*varspec*`)` is required for all `ds`, `po`, and `xpo` commands and for `telasso`.

`xfold(`*#*`)` is required for all `xpo` commands and for `telasso` when the option `xfolds(`*#*`)` was specified.

`resample(`*#*`)` is required for `xpo` and for `telasso` when the option `resample(`*#*`)` was specified.

`tlevel(`*#*`)` is required for the outcome model in `telasso`.

x_unit_spec	Description
`l1norm`	ℓ_1-norm of standardized coefficient vector; the default
`l1normraw`	ℓ_1-norm of unstandardized coefficient vector
`lnlambda`	λ on a logarithmic scale
`rlnlambda`	λ on a reverse logarithmic scale

Options

Main

`xunits(`*x_unit_spec*`)` specifies the x-axis units used for graphing the coefficient paths. The following *x_unit_spec*s are available:

`l1norm` specifies x-axis units ℓ_1-norm of the standardized coefficient vector. This is the default.

`l1normraw` specifies x-axis units ℓ_1-norm of the unstandardized coefficient vector.

`lnlambda` specifies x-axis units λ on a logarithmic scale.

`rlnlambda` specifies x-axis units λ on a reverse logarithmic scale.

`minmax` adds minimum and maximum values to the x axis.

`for(`*varspec*`)` specifies a particular lasso after `telasso` or after a `ds`, `po`, or `xpo` estimation command fit using the option `selection(cv)`, `selection(adaptive)`, or `selection(bic)`. For all commands except `poivregress` and `xpoivregress`, *varspec* is always *varname*.

For the `ds`, `po`, and `xpo` commands except `poivregress` and `xpoivregress`, *varspec* is either *depvar*, the dependent variable, or one of *varsofinterest* for which inference is done.

For `poivregress` and `xpoivregress`, *varspec* is either *varname* or `pred(`*varname*`)`. The lasso for *depvar* is specified with its *varname*. Each of the endogenous variables have two lassos, specified by *varname* and `pred(`*varname*`)`. The exogenous variables of interest each have only one lasso, and it is specified by `pred(`*varname*`)`.

For `telasso`, *varspec* is either the outcome variable or the treatment variable.

This option is required after `telasso` and after the `ds`, `po`, and `xpo` commands.

`xfold(#)` specifies a particular lasso after an `xpo` estimation command or after `telasso` when the option `xfolds(#)` was specified. For each variable to be fit with a lasso, K lassos are done, one for each cross-fit fold, where K is the number of folds. This option specifies which fold, where $\# = 1, 2, \ldots, K$. `xfold(#)` is required after an `xpo` command and after `telasso` when the option `xfolds(#)` was specified.

`resample(#)` specifies a particular lasso after an `xpo` estimation command or after `telasso` fit using the option `resample(#)`. For each variable to be fit with a lasso, $R \times K$ lassos are done, where R is the number of resamples and K is the number of cross-fitting folds. This option specifies which resample, where $\# = 1, 2, \ldots, R$. `resample(#)`, along with `xfold(#)`, is required after an `xpo` command and after `telasso` with resampling.

`tlevel(#)` specifies the lasso for the outcome variable at the specified treatment level after `telasso`. This option is required to refer to the outcome model after `telasso`.

`alpha(#)` graphs coefficient paths for $\alpha = \#$. The default is `alpha(`α^*`)`, where α^* is the selected α. `alpha(#)` may only be specified after `elasticnet`.

`rawcoefs` specifies that unstandardized coefficient paths be graphed. By default, coefficients of standardized variables (mean 0 and standard deviation 1) are graphed.

Reference line

`rlopts(`*cline_options*`)` affects the rendition of the reference line. See [G-3] ***cline_options***.

`norefline` suppresses plotting the reference line.

Path

lineopts(*cline_options*) affects the rendition of all coefficient paths. See [G-3] ***cline_options***. lineopts() is not allowed when there are 100 or more coefficients.

line#opts(*cline_options*) affects the rendition of coefficient path #. See [G-3] ***cline_options***. line#opts() is not allowed when there are 100 or more coefficients.

mono graphs the coefficient paths using a single line. mono is the default when there are 100 or more coefficients in the lasso.

monoopts(*cline_options*) affects the rendition of the line used to graph the coefficient paths when mono is specified. See [G-3] ***cline_options***.

Data

data(*filename* [, replace]) saves the plot data to a Stata data file.

Y axis, X axis, Titles, Legend, Overall

twoway_options are any of the options documented in [G-3] ***twoway_options***, excluding by(). These include options for titling the graph (see [G-3] ***title_options***) and options for saving the graph to disk (see [G-3] ***saving_option***).

Remarks and examples

stata.com

Remarks are presented under the following headings:

Coefficient path plots
An example
Adding a legend
λ scale and reference line
After fitting with sqrtlasso
After fitting with elasticnet
After fitting with inference commands

Coefficient path plots

Coefficient path plots show the path of each coefficient over the search grid for the lasso penalty parameter λ. The grid can be shown as either the log of lambda, xunits(lnlambda); the reverse of that scale, xunits(rlnlambda); the ℓ_1-norm of the standardized coefficients, xunits(l1norm) (the default); or the ℓ_1-norm of the unstandardized coefficients. The ℓ_1-norm of the standardized coefficients is traditionally the default because it directly represents the lasso constraint in the standardized coefficient space—the maximum allowed sum of the absolute values of the coefficients subject to a value of lambda. λ and the ℓ_1-norm have an inverse monotonic relationship. λ is the lasso penalty. The ℓ_1-norm is its impact on the length of the coefficient vector.

Coefficient path plots can be drawn after any command that directly searches over a grid of λ's—that is, after any command that uses option selection(cv), selection(adaptive), or selection(none). They can be drawn after commands lasso, elasticnet, sqrtlasso, or any of the 11 lasso inference commands.

An example

We used the auto dataset to demonstrate the `lasso` command in [LASSO] **lasso**.

```
. sysuse auto
(1978 automobile data)
```

While this dataset is an unlikely candidate for fitting with lasso, it is perfectly good for demonstrating both lasso fitting and `coefpath`.

In that entry, we discussed how to model `mpg` on the remaining covariates in the dataset by typing

```
. lasso linear mpg i.foreign i.rep78 headroom weight turn gear_ratio price
> trunk length displacement, selection(cv, alllambdas) stop(0) rseed(12345)

Evaluating up to 100 lambdas in grid ...
Grid value 1:     lambda =  4.69114   no. of nonzero coef. =       0
  (output omitted)
Grid value 100:   lambda = .0004691   no. of nonzero coef. =      13

10-fold cross-validation with 100 lambdas ...
Fold  1 of 10:  10....20....30....40....50....60....70....80....90....100
  (output omitted)
Fold 10 of 10:  10....20....30....40....50....60....70....80....90....100
... cross-validation complete

Lasso linear model                          No. of obs        =         69
                                            No. of covariates =         15
Selection: Cross-validation                 No. of CV folds   =         10
```

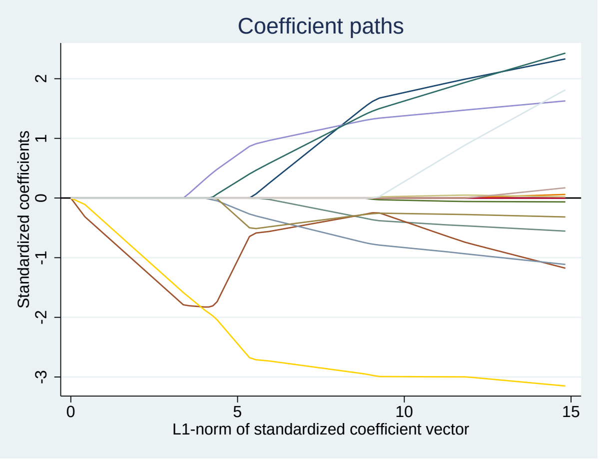
ID	Description	lambda	No. of nonzero coef.	Out-of-sample R-squared	CV mean prediction error
1	first lambda	4.69114	0	0.0049	33.74852
40	lambda before	.1246008	8	0.6225	12.80314
* 41	selected lambda	.1135316	8	0.6226	12.79854
42	lambda after	.1034458	8	0.6218	12.82783
100	last lambda	.0004691	13	0.5734	14.46932

```
* lambda selected by cross-validation.
```

This command is fully explained in [LASSO] **lasso**. Of special interest here is the suboption `alllambdas` and the option `stop(0)`. Together, they ensure that the full 100 default values in the cross-validation grid are searched. Otherwise, `lasso` will stop searching once it has found an optimum or once one of its other stopping rules is met.

Graphing the coefficient paths for this lasso fit is as easy as typing

```
. coefpath
```

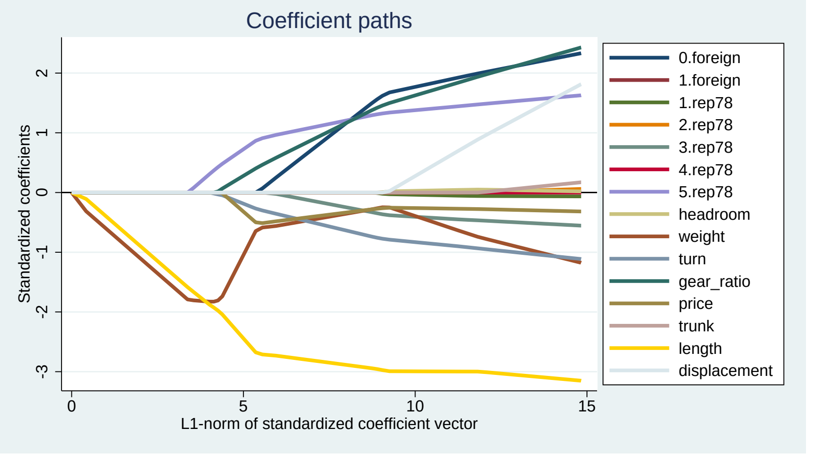


The x axis shows the sum of the absolute values of the penalized coefficients (the ℓ_1-norm) going from 0 to 15. Each line traces the penalized coefficient for one of the standardized covariates in our model. These graphs are popular but pose a bit of a conundrum. They can only be interpreted when there are few covariates, yet lasso is often most applicable when there are many covariates.

Adding a legend

Often, there are too many variables to allow for interest in any single path. These data are small enough that we can look at each covariate. Let's turn the legend on and place it beside the graph, using a single column for the keys,

```
. coefpath, lineopts(lwidth(thick)) legend(on position(3) cols(1)) xsize(4.2)
```



We put the legend on the right of the graph by using the suboption `position(3)` (think position on a clock—3 o'clock). We specified with suboption `cols(1)` that the legend have just one column rather than the default two columns. We requested thicker lines with suboption `lwidth(thick)` to make the paths easier to match to the legend. And, with option `xsize(4.2)`, we requested a slightly wider graph to make room for the legend.

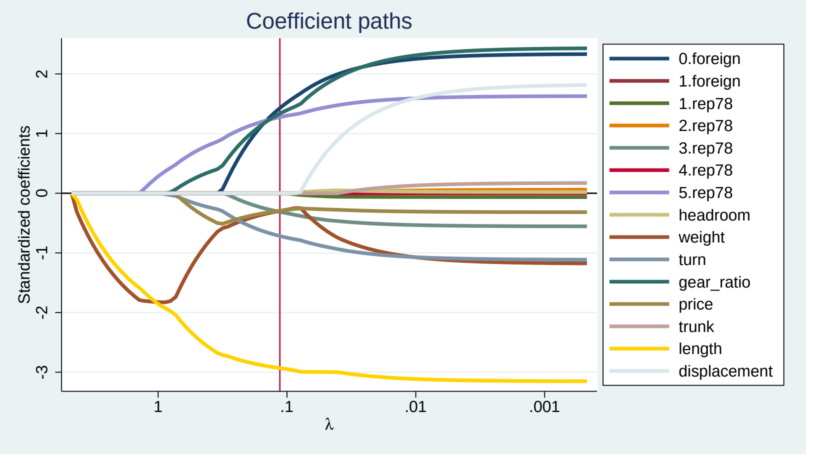
Looking at the graph, we now know which variable is traced by each line. We see that car `weight` is traced by the sienna (reddish brown) line that starts off downward before its effect declines toward 0. What is happening here is that `weight` enters early and absorbs any effect of other variables that are correlated with it but have yet to enter the model. When `5.rep78` enters the model, the coefficient on `weight` flattens. As `gear_ratio`, `price`, and `turn` enter, the effect of `weight` is further attenuated toward 0. This is simply what happens when correlated variables are added to a model. With lasso, they are added slowly because the lasso penalty brings in the coefficients in a penalized form rather than all at once.

Lasso is just letting variables into the model based on its penalty and the current value of lambda. We can see what is happening, but that is about it.

λ scale and reference line

In this example from [LASSO] **lasso**, we might find it yet more interesting to put our plot on the same scale as the `cvplot` from that entry and add a reference line for the λ selected by cross-validation. We change the scale by adding `xunits(rlnlambda)` and place the reference line by adding `xline(.1135)`,

```
. coefpath, lineopts(lwidth(thick)) legend(on position(3) cols(1)) xsize(4.2)
> xunits(rlnlambda) xline(.1135)
```



We know from the output of `lasso` that cross-validation selected eight coefficients. We can now see where each of them is in its path when cross-validation selected a model.

After fitting with sqrtlasso

There is not much to say about using `coefpath` after fitting with `sqrtlasso`. You type the same thing after `sqrtlasso` that you would type after `lasso`.

If you wish to see that, you can simply change `lasso` to `sqrtlasso` in the estimation command above. Make no changes to any other commands.

What's more, you can add the option `sqrtlasso` whenever it is allowed to any of the inference commands below. Nothing changes in the way we specify our `coefpath` commands.

After fitting with elasticnet

The only thing that changes with `coefpath` after an `elasticnet` command is that we can specify the option `alpha()` to graph the paths for a value of α that is different than the alpha chosen by `elasticnet`.

We can fit an `elasticnet` model using the auto dataset:

```
. elasticnet linear mpg i.foreign i.rep78 headroom weight turn gear_ratio
> price trunk length displacement,
> selection(cv, alllambdas) stop(0) rseed(12345)
```

(output omitted)

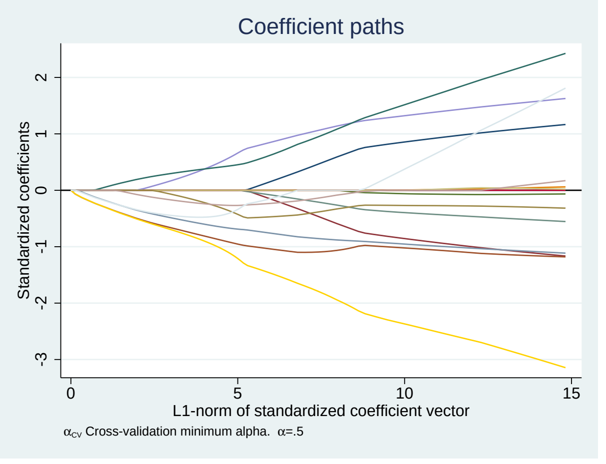
```
Elastic net linear model                     No. of obs         =         69
                                             No. of covariates =         15
Selection: Cross-validation                  No. of CV folds    =         10
```

alpha	ID	Description	lambda	No. of nonzero coef.	Out-of-sample R-squared	CV mean prediction error
1.000						
	1	first lambda	9.382281	0	-0.0064	34.13399
	109	last lambda	.0004691	13	0.5734	14.46932
0.750						
	110	first lambda	9.382281	0	-0.0064	34.13399
	218	last lambda	.0004691	14	0.5736	14.46276
0.500						
	219	first lambda	9.382281	0	-0.0033	34.02853
	264	lambda before	.1647149	11	0.6328	12.45289
	* 265	selected lambda	.1500821	11	0.6331	12.44435
	266	lambda after	.1367492	11	0.6331	12.44506
	327	last lambda	.0004691	14	0.5738	14.4564

* alpha and lambda selected by cross-validation.

We see that cross-validation chose α to be 0.5. Had it chosen 1, the `elasticnet` would have reduced to `lasso`. To see the coefficient path graph for $\alpha = 0.5$, we simply type

```
. coefpath
```



That looks quite a bit different from the first graph we drew in this entry, which is the graph for lasso and would be the same as the graph we would get if we added the option `alpha(1)`.

If we wanted the graph for $\alpha = 0.75$, we would type

```
. coefpath, alpha(.75)
```

After fitting with inference commands

All postestimation tools, including `coefpath`, can be used after the `ds`, `ps`, and `xpo` inference commands. Of all the postestimation commands, `coefpath` is the least likely to be useful in this context. The inference commands use lassos to select control variables from a set of potential controls. Aside from diagnosing whether something pathological occurred in the lasso, you are not supposed to care which controls were selected, much less their coefficients, and even less the path of those coefficients. Regardless, you can draw coefficient path plots for any lasso run by an inference command.

We will use a few of the examples from [LASSO] **Inference examples** to show you what to type to create a coefficient path plot.

All these examples use `breathe.dta`, which attempts to measure the effect of nitrogen dioxide on the reaction time of school children. All these examples will run, but we dispense with the output here. If you are curious, run some.

To prepare the dataset, type

```
. use https://www.stata-press.com/data/r17/breathe
. do no2
```

All the `ds` (double-selection) and `po` (partialing-out) `coefpaths` are drawn in exactly the same way. To fit one of the double-selection models from [LASSO] **Inference examples**, we type

```
. dsregress react no2_class, controls($cc i.($fc)) selection(cv) rseed(12345)
```

Recall that we are using global macros `$cc` and `$fc` to hold our control variables. `$cc` holds the continuous controls, and `$fc` holds the factor-variable controls. Typing `$cc` simply substitutes the list of continuous controls into our command, and likewise for `$fc`. We write `i.($fc)` so that each of the variables in `$fc` is expanded into dummy variables for each distinct level of the variable.

To draw the coefficient path plot for the lasso of the dependent variable `react`, we type

```
. coefpath, for(react)
```

To draw the plot for the lasso of the variable of interest `no2_class`, we type

```
. coefpath, for(no2_class)
```

If we had fit the models via partialing out by typing `poregress` instead of `dsregress`, nothing would change. Typing `coefpath, for(react)` would still produce the coefficient path plot for the lasso of `react`, and typing `coefpath, for(no2_class)` would still produce the plot for `no2_class`.

What's more, what we type to plot coefficient paths does not change if our dependent variable were dichotomous and we had fit the model by using `dslogit` or `pologit`. Nor does it change if the dependent variable is a count and we fit the model by using `dspoisson` or `popoisson`.

Things do change if we fit the model by using the `xpo` (cross-fit partialing-out) estimators. The `xpo` estimators perform lots of lassos. Let's refit our original model using `xporegress`.

```
. xporegress react no2_class, controls($cc i.($fc)) selection(cv) rseed(12345)
  (output omitted )
```

To see the lassos that `xporegress` ran, we can use `lassoinfo`:

```
. lassoinfo, each

    Estimate: active
     Command: xporegress
```

Dependent variable	Model	Selection method	xfold no.	Selection criterion	lambda	No. of selected variables
no2_class	linear	cv	1	CV min.	.1801304	14
no2_class	linear	cv	2	CV min.	.2561599	10
no2_class	linear	cv	3	CV min.	.2181624	13
no2_class	linear	cv	4	CV min.	.1963854	13
no2_class	linear	cv	5	CV min.	.2352711	11
no2_class	linear	cv	6	CV min.	.2663564	12
no2_class	linear	cv	7	CV min.	.1293717	16
no2_class	linear	cv	8	CV min.	.1722497	15
no2_class	linear	cv	9	CV min.	.264197	9
no2_class	linear	cv	10	CV min.	.1184878	16
react	linear	cv	1	CV min.	2.130811	19
react	linear	cv	2	CV min.	2.443412	16
react	linear	cv	3	CV min.	2.062956	17
react	linear	cv	4	CV min.	4.220311	13
react	linear	cv	5	CV min.	7.434224	8
react	linear	cv	6	CV min.	3.356193	14
react	linear	cv	7	CV min.	7.954354	6
react	linear	cv	8	CV min.	6.422852	8
react	linear	cv	9	CV min.	2.982171	15
react	linear	cv	10	CV min.	2.738883	18

That's 20 lassos! `react` has 10 and `no2_class` has 10. There is one lasso for each variable for each cross-validation fold. The cross-validation folds are enumerated in the column titled `xfold no.`. To see the cross-validation plot for the third cross-validation fold for the variable `react`, we type

```
. coefpath, for(react) xfold(3)
```

Change `react` to `no2_class` to see the plot for `no2_class`.

Feel free to plot all 18 other pairings of each variable with the cross-validation folds.

Again, it would not matter if we had fit `xpologit` or `xpopoisson` models. We type the same thing to see our coefficient path plots.

The cross-fit models can create even more lassos. We are willing to resample the whole process to reduce the sampling variability. Let's resample the process 10 times:

```
. xporegress react no2_class, controls($cc i.($fc)) selection(cv)   ///
      resample(10) rseed(12345)
```

If you type that command, be patient; it takes a few minutes to run.

Now, let's look at our lassos:

```
. lassoinfo, each
    Estimate: active
     Command: xporegress
```

Dependent variable	Model	Selection method	Resample number	xfold no.	Selection criterion	lambda	No. of sel. var.
no2_class	linear	cv	1	1	CV min.	.1801304	14
no2_class	linear	cv	1	2	CV min.	.2561599	10
(output omitted)							
no2_class	linear	cv	1	10	CV min.	.1184878	16
no2_class	linear	cv	2	1	CV min.	.2118238	12
(output omitted)							
no2_class	linear	cv	2	10	CV min.	.1773874	13
(output omitted)							
no2_class	linear	cv	3	10	CV min.	.1676957	13
react	linear	cv	1	1	CV min.	2.130811	19
(output omitted)							
react	linear	cv	1	10	CV min.	2.738883	18
react	linear	cv	2	1	CV min.	4.379673	14
(output omitted)							
react	linear	cv	2	10	CV min.	3.747121	14
react	linear	cv	3	1	CV min.	5.821677	11
(output omitted)							
react	linear	cv	3	10	CV min.	3.668243	13

We now have 30 of them! There is one for each variable within each cross-validation sample within each resample sample. Here is how we would graph the coefficient path plot for the third cross-validation sample in the second resample sample for the covariate of interest `no2_class`.

```
. coefpath, for(no2_class) resample(2) xfold(3)
```

If we had typed `resample(10)` instead of `resample(3)` on our `xporegress` command, we would have 200 possible graphs. Have fun looking at those.

Yet again, it would not matter if we had fit `xpologit` or `xpopoisson` models. We still type the same thing to see our coefficient path plots.

Also see

[LASSO] **lasso postestimation** — Postestimation tools for lasso for prediction

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

[TE] **telasso postestimation** — Postestimation tools for telasso