

[Description](#)[Remarks and examples](#)[References](#)[Also see](#)

Description

Lasso for inference comprises 11 related estimation commands and several postestimation commands for performing inference about a true model. Fitting and interpreting inferential lasso models is demonstrated via examples.

Remarks and examples

stata.com

Remarks are presented under the following major headings:

- [1 Overview](#)
- [2 Fitting and interpreting inferential models](#)
- [3 Fitting logit inferential models to binary outcomes. What is different?](#)
- [4 Fitting inferential models to count outcomes. What is different?](#)
- [5 Exploring inferential model lassos](#)
- [6 Fitting an inferential model with endogenous covariates](#)

1 Overview

1.1 How to read the example entries

All the examples demonstrate something about the inferential lasso models, so we obviously think you should read this entire section. That said, there are a lot of pages, so here are some other options.

Everyone should read [1.3 Review of concepts](#), [2.1 Overview of inferential estimation methods](#), and [2.2 Fitting via cross-fit partialing out \(xpo\) using plugin](#). What you read there is essential to using and understanding all the inferential models. We are pretty sure you will also want to read [2.3 Fitting via cross-fit partialing out \(xpo\) using cross-validation](#), [2.4 Fitting via double selection \(ds\) using cross-validation](#), and [2.5 Fitting via the other 22 methods](#).

We use the variable-management tool `v1` to manage the variable lists used in all the examples, and most of the examples use a common dataset. We introduce both in [1.4 The primary dataset](#). We say enough in sections 2.1 and 2.2 that you will not be lost if you do not read section 1.4. But you will better understand the dataset—and how we are manipulating it—if you read section 1.4.

If you are only interested in logit models for binary outcomes, then [3 Fitting logit inferential models to binary outcomes. What is different?](#) is essential reading, but only after reading sections 1.3, 2.1, and 2.2. Similarly, if your sole interest is Poisson models for count outcomes, then read [4 Fitting inferential models to count outcomes. What is different?](#), but only after reading sections 1.3, 2.1, and 2.2.

The titles on all other sections are relatively self explanatory. So, if you are not reading all the sections, choose from them based on your interest.

1.2 Detailed outline of the topics

- 1 Overview
 - 1.1 How to read the example entries
 - 1.2 Detailed outline of the topics
 - 1.3 Review of concepts
 - 1.4 The primary dataset
- 2 Fitting and interpreting inferential models
 - 2.1 Overview of inferential estimation methods
 - 2.2 Fitting via cross-fit partialing out (xpo) using plugin
 - 2.3 Fitting via cross-fit partialing out (xpo) using cross-validation
 - 2.4 Fitting via double selection (ds) using cross-validation
 - 2.5 Fitting via the other 22 methods
 - 2.6 Fitting models with several variables of interest
 - 2.7 Fitting models with factor variables of interest
 - 2.8 Fitting models with interactions of interest
 - 2.9 Fitting models with a nonlinear relationship of interest
 - 2.10 Controls are controls
- 3 Fitting logit inferential models to binary outcomes. What is different?
 - 3.1 Interpreting standard odds ratios
 - 3.2 Interpreting models with factor variables, nonlinear relationships, and interactions
- 4 Fitting inferential models to count outcomes. What is different?
 - 4.1 Interpreting standard incidence-rate ratios
 - 4.2 Interpreting models with factor variables
- 5 Exploring inferential model lassos
- 6 Fitting an inferential model with endogenous covariates

1.3 Review of concepts

We have said a lot about the inferential estimation commands elsewhere in this manual. For a quick overview that describes what you need to know, and just what you need to know, see [\[LASSO\] Lasso intro](#). For a deeper understanding of lasso for inference, read [\[LASSO\] Lasso inference intro](#). We highly recommend reading both of those sections.

The inferential lasso estimators require you to break up your model into two parts: the part about which you need to perform inference and the part about which you do not care. Let's call the first part the “inference part” and the second part the “noninference part”.

Often, the inference part is a single variable, perhaps even a single indicator variable, such as “walks at least three miles a week”. The inference part could be more complicated than a single variable. It might involve several variables, polynomials, or interactions. But, it will generally be relatively small.

The noninference part can be much larger. What you include there will sometimes reflect an ignorance of how that part relates to your outcome. Often, our theory or intuition involves only a few variables, our variables of interest. We know lots of other things affect our outcome; we just have little or no guidance about which things are important or how they relate to our outcome. We will call the variables in this noninference part controls. What makes lasso for inference special is that you need not understand how those controls affect the outcome.

There are other requirements. We said that the inference part will typically be small. The number of controls that lasso needs to include must also be small with respect to the sample size. See [Solutions that focus on the true model](#) in [\[LASSO\] Lasso inference intro](#).

1.4 The primary dataset

To demonstrate the inference commands, we will mostly use one dataset—a real-world dataset that includes children’s performance on a test of reaction time, levels of nitrogen dioxide (NO₂) pollution, the children’s physical and socioeconomic characteristics, and some other environmental factors. The data were collected and analyzed by [Sunyer et al. \(2017\)](#).

Our interest is in how levels of nitrogen dioxide in the classroom affect the children’s performance on the test, while adjusting for other factors. We will focus on two outcomes from the Attention Network Test (ANT)—reaction time and omissions. For linear models, we will use hit reaction time—a measure of speed in responding to stimuli. For Poisson models, we will use omissions—the number of times the child failed to respond to a stimulus. For logit models, we will use whether there were any omissions.

We are using an extract of the data and focusing on how to use the software, so let’s not get ideas about publishing any of this.

Let’s take a quick look at the dataset.

```
. use https://www.stata-press.com/data/r18/breathe
(Nitrogen dioxide and attention)

. describe

Contains data from https://www.stata-press.com/data/r18/breathe.dta
Observations:      1,089          Nitrogen dioxide and attention
Variables:         22           21 Jun 2022 12:43
                          (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
react	double	%10.0g		* Reaction time (ms)
correct	int	%10.0g		* Number of correct responses
omissions	byte	%10.0g		* Failure to respond to stimulus
no2_class	float	%9.0g		Classroom NO2 levels (ug/m3)
no2_home	float	%9.0g		Home NO2 levels (ug/m3)
age	float	%9.0g		Age (years)
age0	double	%4.1f		Age started school
sex	byte	%9.0g	sex	Sex
grade	byte	%9.0g	grade	Grade in school
overweight	byte	%32.0g	overwt	* Overweight by WHO/CDC definition
lbweight	byte	%18.0g	lowbw	* Low birthweight
breastfeed	byte	%19.0f	bfeed	Duration of breastfeeding
msmoke	byte	%10.0f	smoke	* Mother smoked during pregnancy
meducation	byte	%17.0g	edu	Mother’s education level
feducation	byte	%17.0g	edu	Father’s education level
siblings_old	byte	%1.0f		Number of older siblings in house
siblings_young	byte	%1.0f		Number of younger siblings in house
sev_home	float	%9.0g		Home socio-economic vulnerability index
green_home	double	%10.0g		Home greenness (NDVI), 300m buffer
noise_school	float	%9.0g		School noise levels (dB)
sev_school	float	%9.0g		School socio-economic vulnerability index
precip	double	%10.0g		Daily total precipitation
				* indicated variables have notes

Sorted by:

This is not a large dataset, just 22 variables. Regardless, we are going to use the `vl` tools to create the variable lists we need for our analysis. This may seem like a detour, but `vl` is useful even for small datasets, and it is nearly indispensable if your dataset has hundreds or even tens of thousands of variables.

Our goal is to create two lists of control covariates, for example, independent variables. One list will contain continuous control covariates and the other will contain categorical control covariates. Why not just one list? Because we want the categorical variables to enter our model as indicator variables for each level (distinct value) of the categorical variable. To expand a categorical variable into indicator variables for its levels, we must prefix it with an `i.`, for example, `i.grade`.

Starting with `vl` is easy: we just type `vl set`,

```
. vl set
```

Macro	Macro's contents	
	# Vars	Description
System		
\$vlcategorical	10	categorical variables
\$vlcontinuous	10	continuous variables
\$vluncertain	2	perhaps continuous, perhaps categorical variables
\$vlother	0	all missing or constant variables

Notes

1. Review contents of `vlcategorical` and `vlcontinuous` to ensure they are correct. Type `vl list vlcategorical` and type `vl list vlcontinuous`.
2. If there are any variables in `vluncertain`, you can reallocate them to `vlcategorical`, `vlcontinuous`, or `vlother`. Type `vl list vluncertain`.
3. Use `vl move` to move variables among classifications. For example, type `vl move (x50 x80) vlcontinuous` to move variables `x50` and `x80` to the continuous classification.
4. `vl` names are global macros. Type the `vl` name without the leading dollar sign (\$) when using `vl` commands. Example: `vlcategorical` not `$vlcategorical`. Type the dollar sign with other Stata commands to get a `varlist`.

`vl` has divided our 22 variables into 4 groups and placed those groups into [global macros](#). Do not worry about the technical term global macro. Just know that once the macro `$vlcategorical` has been created, any time you type `$vlcategorical`, you will get the full list of categorical variables.

```
. display "$vlcategorical"
sex grade overweight lbweight breastfeed msmoke meducation feducation
> siblings_old siblings_young
```

That is convenient! `vl` has placed all of our categorical variables into one bin called `$vlcategorical`. Now let's follow the instructions in the notes after `vl` to be sure we like what `vl` did on our behalf.

```
. vl list vlcategorical
```

Variable	Macro	Values	Levels
sex	\$vlcategorical	0 and 1	2
grade	\$vlcategorical	integers >=0	3
overweight	\$vlcategorical	0 and 1	2
lbweight	\$vlcategorical	0 and 1	2
breastfeed	\$vlcategorical	integers >=0	3
msmoke	\$vlcategorical	0 and 1	2
meducation	\$vlcategorical	integers >=0	4
feducation	\$vlcategorical	integers >=0	4
siblings_old	\$vlcategorical	integers >=0	5
siblings_young	\$vlcategorical	integers >=0	5

Among other things, we see that `sex` has just two values, 0 and 1; and `meducation` (mother's education level) has four values that are integers greater than or equal to 0.

Usually with categorical variables, we intend to create indicator variables for each unique value (level) the variable takes on. So we are looking for variables that do not fit that purpose. `siblings_old` and `siblings_young` have five values, but even their names make one think they might be counts. Let's look further at `siblings_old`:

```
. tabulate siblings_old
```

Number of older siblings in house	Freq.	Percent	Cum.
0	564	52.17	52.17
1	424	39.22	91.40
2	84	7.77	99.17
3	8	0.74	99.91
4	1	0.09	100.00
Total	1,081	100.00	

It does look like a count of siblings. We might want indicators for each count (level), or we might want it to enter our model linearly as a continuous variable. That singleton count of 4 older siblings will have to be dropped whenever we perform cross-validation or cross-fitting because it cannot be in both the estimation and the validation samples. We could recode the values to represent 0, 1, 2, and 3-or-more siblings and keep it a factor variable. After all, lasso is a technique built for handling lots of variables. It is easier for our examples to simply redesignate the two counts of siblings as continuous:

```
. vl move (siblings_old siblings_young) vlcontinuous
note: 2 variables specified and 2 variables moved.
```

Macro	# Added/Removed
\$vlcategorical	-2
\$vlcontinuous	2
\$vluncertain	0
\$vlother	0

Let's now take a look at all variables designated continuous. We will use `summarize` to get a bit more detail:

```
. summarize $vlcontinuous
```

Variable	Obs	Mean	Std. dev.	Min	Max
react	1,084	742.4808	145.4446	434.0714	1303.26
no2_class	1,089	30.16779	9.895886	7.794096	52.56397
no2_home	1,089	54.71832	18.04786	2.076335	118.6568
age	1,089	9.08788	.886907	7.45243	11.63313
age0	1,082	3.218022	1.293168	0	9
sev_home	1,089	.4196807	.1999143	.0645161	.9677419
green_home	1,089	.1980721	.077777	.0184283	.5258679
noise_school	1,089	37.96354	4.491651	28.8	51.1
sev_school	1,089	.4096389	.2064394	.1290323	.8387097
precip	1,089	.5593205	1.2364	0	5.8
siblings_old	1,081	.573543	.6752252	0	4
siblings_y~g	1,083	.565097	.6906831	0	6

We notice three things. First, `age0` has a min of 0 and a max of 9—both integers. Did `v1 set` make a mistake? Let's look carefully:

```
. tabulate age0
```

Age started school	Freq.	Percent	Cum.
0.0	4	0.37	0.37
0.3	1	0.09	0.46
0.5	1	0.09	0.55
0.8	1	0.09	0.65
0.9	1	0.09	0.74
1.0	33	3.05	3.79
1.4	1	0.09	3.88
1.5	8	0.74	4.62
1.7	1	0.09	4.71
2.0	116	10.72	15.43
2.5	4	0.37	15.80
2.8	3	0.28	16.08
2.9	1	0.09	16.17
3.0	739	68.30	84.47
4.0	35	3.23	87.71
5.0	39	3.60	91.31
6.0	54	4.99	96.30
7.0	25	2.31	98.61
8.0	8	0.74	99.35
9.0	7	0.65	100.00
Total	1,082	100.00	

No mistake. There are fractional values. Also, looking back at the results of our `describe`, we see that `age0` is the age at which the students started school. We do want to treat that as continuous.

Second, our dependent variable, `react`, is in the list. It is continuous, and so it belongs there. However, we will need to take care that we do not include it among our control covariates.

Third, our covariate of interest, `no2_class`, is also in the list. As with `react`, we will need to exclude it from the control covariates.

What of those two variables that were in `$vluncertain`?

```
. vl list vluncertain
```

Variable	Macro	Values	Levels
correct	<code>\$vluncertain</code>	integers >=0	41
omissions	<code>\$vluncertain</code>	integers >=0	27

`vl set` knows they are integer, but one has 41 distinct values and the other has 27. `vl` is unwilling to classify them as either continuous or categorical. See [D] `vl` for how to change `vl`'s rules. We said earlier that `omissions` is another outcome variable from the ANT. `correct` is also an outcome variable from the ANT. Both are potential dependent variables, meaning that neither are valid controls. We will leave them where they are.

We were fortunate that `correct` and `omissions` were already left out of `$vlcategorical` and `$vlcontinuous`. Otherwise, it would be our job to ensure they are not included among the controls. `vl` is convenient for classifying variables, but it does not truly understand anything about the meaning of the variables. It is our job to know which variables are actually other outcomes or transformations of the outcomes.


Let's now create our own two global macros. One will have continuous covariates, and we will call that macro `cc`. The other will have categorical covariates, which we will treat as factor covariates, and we will call that macro `fc`.

```
. vl create cc = vlcontinuous - (react no2_class)
note: $cc initialized with 10 variables.

. vl create fc = vlcategorical
note: $fc initialized with 8 variables.
```

`fc` is just a copy of `vlcategorical`. We could just use `vlcategorical`, but it is best to create our own macro in case we want to change it later. When we created `cc`, we removed our dependent variable, `react`, and covariate of interest, `no2_class`. That gives us a list of continuous controls.

Now we have control covariate lists we can use in our inference commands.

No one at StataCorp would ever type everything we just typed interactively. We would open an editor or the Do-file Editor and work there. I suggest you do the same thing. Click on the Do-file Editor button, . Then type in the Editor

```
describe
vl set
vl list vlcategorical
tabulate siblings_old
vl move (siblings_old siblings_young) vlcontinuous
summarize $vlcontinuous
tabulate age0
vl list vluncertain
vl create cc = vlcontinuous - (react no2_class)
vl create fc = vlcategorical
```

Save the file as `no2.do`. Then you can type `do no2` to re-create your control covariate lists.

If you want to exclude the exploratory commands, just type

```
vl set
vl move (siblings_old siblings_young) vlcontinuous
vl create cc = vlcontinuous - (react no2_class)
vl create fc = vlcategorical
```

2 Fitting and interpreting inferential models

2.1 Overview of inferential estimation methods

Considering only the linear models for continuous outcomes and ignoring endogeneity, there are 25 methods to fit any given model. There are three commands—`dsregress`, `poregress`, and `xporegress`. The `po` and `xpo` commands allow the option `semi`, which adjusts how they partial out, making five methods. Within each of these methods, there is an option allowing three ways of selecting the lasso penalty λ —`selection(plugin)`, `selection(cv)`, and `selection(adaptive)`. And, for 10 of these 15 methods, there is an option (`sqrtlasso`) to specify that the square-root lasso rather than the standard lasso be used to select covariates. Square-root lasso cannot be combined with `selection(adaptive)`.

What you type differs only a little when requesting any of these 25 methods. More importantly, you interpret the coefficient estimates, standard errors, and confidence intervals exactly the same across all 25 methods. Which is to say, you interpret them exactly as you would interpret the estimates from linear regression.

Let's see how to request each of these 25 methods.

Assume that our dependent variable is y . We will include two covariates of interest— d_1 and d_2 . We will specify 100 potential continuous control covariates— x_1 – x_{100} . And, we have 30 potential factor control variables— f_1 – f_{30} . The factor variables could be ordered, unordered, or just indicators. We specify them as `i.(f1-f30)` so that each level of each covariate is included as its own term. So, if f_3 has four levels, then it introduces four indicator variables (covariates) into the potential controls. See [U] 11.4.3 **Factor variables**. We could also introduce interactions among the factor variables, among the continuous variables, or both. Do that if you wish.

All these commands will run if you use `lassoex.dta`.

To make the commands easier to read, we do not specify option `rseed()` to make reproducible the commands that randomly split the samples repeatedly. If you want the results to be the same each time you run the commands, add `rseed(12345)` (or whatever number you like).

```
. use https://www.stata-press.com/data/r18/lassoex
```

We can first fit the model using the cross-fit partialing-out method, the partialing-out method, and the double-selection method. In all cases, we are using the default plugin method for choosing the included controls via its choice of the lasso penalty parameter λ .

```
. xporegress y d1 d2, controls(x1-x100 i.(f1-f30))
. poregress y d1 d2, controls(x1-x100 i.(f1-f30))
. dsregress y d1 d2, controls(x1-x100 i.(f1-f30))
```

We can fit the same models, but this time using the cross-validation method to choose the lasso penalty parameter λ and thereby to choose the included control covariates.

```
. xporegress y d1 d2, controls(x1-x100 i.(f1-f30)) selection(cv)
. poregress y d1 d2, controls(x1-x100 i.(f1-f30)) selection(cv)
. dsregress y d1 d2, controls(x1-x100 i.(f1-f30)) selection(cv)
```

Again, we can fit the same models, but this time using the adaptive method to choose the included control covariates.

```
. xporegress y d1 d2, controls(x1-x100 i.(f1-f30)) selection(adaptive)
. poregress y d1 d2, controls(x1-x100 i.(f1-f30)) selection(adaptive)
. dsregress y d1 d2, controls(x1-x100 i.(f1-f30)) selection(adaptive)
```


We can rerun each of the first six methods using the square-root lasso rather than the standard lasso, by adding the option `sqrtlasso`. Here is one example that uses the cross-fit partialing-out method with plugin selection:

```
. xporegress y d1 d2, controls(x1-x100 i.(f1-f30)) sqrtlasso
```

And, we can rerun any of the 10 methods that use commands `poregress` or `xporegress`, including those with `sqrtlasso`, using the `semi` option to specify an alternate form of partialing out. Here is one example:

```
. xporegress y d1 d2, controls(x1-x100 i.(f1-f30)) semi
```

We apologize for the bewildering array of choices. Lasso and machine learning is an active area of research, and you may want the flexibility to choose among these options. That said, if your interest is in your research and not in researching lasso, we feel reasonably comfortable making some suggestions based on the state of the lasso literature at the time this manual was written.

1. Use `xporegress` with no options to fit your model using the cross-fit partialing-out method with λ , and thereby the control covariates, selected using the plugin method.

The plugin method was designed for variable selection in this inferential framework and has the strongest theoretical justification.

2. If you want to explore the process whereby the control covariates were selected, add option `selection(cv)` to your `xporegress` specification.

You can then explore the path by which each lasso selected control covariates.

You are still on firm theoretical footing. Cross-validation meets the requirements of a sufficient variable-selection method.

Cross-validation has a long history in machine learning. Moreover, what cross-validation is doing and how it chooses the covariates is easy to explain.

3. If you do not want to explore lots of lassos and you want to fit models much more quickly, use commands `dsregress` or `poregress` rather than using `xporegress`.

`xporegress` fits 10 lassos for the dependent variable and 10 more lassos for each covariate of interest! That is the default; you can request more. Or you can request fewer, but that is not recommended. So, `xporegress` is orders of magnitude slower than `poregress` and `dsregress`. And it has orders of magnitude more lassos to explore. Overwhelming.

Why then is `xporegress` our first recommendation? It is safer if you think that the process that generated your data has lots of covariates relative to your sample size. Similarly, it is also safer if you want to explore lots of potential controls. The number of potential controls is not as problematic as the number of true covariates because it is the natural log of the potential control that counts. For example, needing 10 additional true covariates is the same as requesting just over 22,000 potential controls. The jargon term for this is sparsity. `xporegress` has a weaker sparsity requirement than do `poregress` and `dsregress`. See [Solutions that focus on the true model](#) in [\[LASSO\] Lasso inference intro](#).

Despite this benefit, if your model is weakly identified by the data, `dsregress` can be more stable than either `poregress` or `xporegress`. `dsregress` uses a union of all the selected controls from all the lassos for all of its computations after selection. Both `poregress` and `xporegress` use the results of each lasso separately to perform parts of their computations (specifically, to compute their moments), and then put all that together when solving the moment conditions. This makes `poregress` and `xporegress` sensitive to which controls are selected for each lasso. So if you change your specification slightly, `dsregress` may be more stable. To be clear, we said more stable, not better.

4. We have suggested `xporegress` without a selection option and `xporegress`, `poregress`, and `dsregress` with option `selection(cv)`. Feel free to try any of the remaining 21 methods. They all meet the requirements of sufficient variable-selection methods, so all can be theoretically justified.

Everything we said above applies to models for binary outcomes fit using `xpologit`, `pologit`, and `dslogit`; and it applies to models for count outcomes fit using `xpopoisson`, `popoisson`, and `dsipoisson`.

These suggestions are based on the assumption that you are not concerned that you have violated or are near the method’s sparsity bound. See *Solutions that focus on the true model* in [LASSO] **Lasso inference intro** for a discussion of sparsity bounds. Data that fit your model poorly can trigger a sparsity bound sooner than data that fit well. If you are concerned, see some alternate but similar suggestions in [LASSO] **Inference requirements**.

2.2 Fitting via cross-fit partialing out (xpo) using plugin

In the previous section, we recommended using the cross-fit partialing-out estimator `xporegress` as your first option. We will use that method to fit a model of how levels of nitrogen dioxide (`no2_class`) in a classroom affect the reaction time (`react`) of students. We use the dataset described in section 1.4.

```
. use https://www.stata-press.com/data/r18/breathe, clear
(Nitrogen dioxide and attention)
```

We created a do-file in section 1.4 that collects our variables into groups that are convenient for specifying inferential lasso models. If you have it saved, great. We will run the one from the Stata Press website:

```
. do https://www.stata-press.com/data/r18/no2
(output omitted)
```

Recall that the purpose of the inferential lasso estimators is to estimate the relationship between one, or a few, covariates of interest and a dependent variable, while adjusting for a possibly large set of control variables. And by “large”, we mean perhaps many more controls than you have observations.

We now have our list of continuous control variables in global macro `$cc` and our list of factor-variable control variables in global macro `$fc`. What does that mean? Anywhere we type `$cc`, Stata substitutes the list of continuous controls, and anywhere we type `$fc`, Stata substitutes the list of factor controls. Let’s display them:

```
. display "$cc"
no2_home age age0 sev_home green_home noise_school sev_school precip siblings_o
> ld siblings_young
. display "$fc"
sex grade overweight lbweight breastfeed msmoke medication feducation
```

That is going to save us a lot of typing.

Now we are ready to fit our model.

```
. xporegress react no2_class, controls($cc i.($fc)) rseed(12345)
Cross-fit fold 1 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 2 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 3 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 4 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 5 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 6 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 7 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 8 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 9 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit fold 10 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for no2_class using plugin
Cross-fit partialing-out
linear model      Number of obs           =      1,036
                  Number of controls          =        32
                  Number of selected controls =        10
                  Number of folds in cross-fit =        10
                  Number of resamples         =         1
                  Wald chi2(1)                =       22.87
                  Prob > chi2                 =       0.0000
```

react	Robust		z	P> z	[95% conf. interval]	
	Coefficient	std. err.				
no2_class	2.316063	.4843097	4.78	0.000	1.366834	3.265293

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The construct `i.($fc)` in `controls()` is [factor-variable](#) notation that expands each variable in `$fc` into indicator variables for each distinct value of the variable. We specified `rseed(12345)` to set the seed of the random-number generator so that our results are reproducible. We did this because the cross-fit estimator uses cross-fitting and so divides the sample into random groups. If we do not set the seed, we will get slightly different results each time we run the command. There is nothing special about 12345; choose any number you like. You will get different, but hopefully similar, results for any seed. The same seed will always produce the same results.

Now to the output. That is a long log. `xporegress` is just reporting on its progress as it performs 10 cross-fits and then performs 2 lassos within each group. We see in the header that 1,036 observations were used, that we specified 32 controls, that 10 controls were selected from the 32, and that we did not resample. From the Wald statistic and its p -value, we see that our covariate of interest is highly significant.

We interpret the coefficient estimates just as we would for a standard linear regression. Because this is linear regression, that effect can be interpreted as the population average effect, the effect for any individual, or the effect for any group. What we lose with the inferential lasso estimators is the ability to interpret any other coefficients.

Our point estimate for the effect of nitrogen dioxide on reaction time is 2.3, meaning that we expect reaction time to go up by 2.3 milliseconds for each microgram per cubic meter increase in nitrogen dioxide. This value is statistically different from 0 well beyond the 5% level, in fact, beyond the 0.1% level. Our 95% confidence interval is 1.4 to 3.3.

We also note that `xporegress` estimates robust standard errors, so all the associated statistics are also robust. With `xporegress`, we are robust to nonnormality of the error and to heteroskedasticity.

We can see how stable the lasso selection of controls is by typing `lassoinfo`.

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

Variable	Model	Selection method	No. of selected variables		
			min	median	max
<code>no2_class</code>	linear	plugin	5	5	5
<code>react</code>	linear	plugin	3	5	5

We see that, over the 10 cross-fits, the plugin method selected 5 controls for the lasso on the covariate of interest—`no2_class`. It selected 5 controls every time. For the dependent variable, `react`, the plugin method selected between 3 and 5 controls. Even though these are real data, they look to be easy for the lasso and plugin to handle. There is nothing to interpret in this table, though if some of the lassos are consistently selecting 0 controls, you might want to explore further. See [Solutions that focus on the true model](#) in [\[LASSO\] Lasso inference intro](#) and see [\[LASSO\] Inference requirements](#).

2.3 Fitting via cross-fit partialing out (xpo) using cross-validation

Continuing with the example above, we can use cross-validation to select our controls rather than plugin. Cross-validation is a well-established method in the machine-learning literature. Even so, it is known to select more variables than are absolutely necessary. We add `selection(cv)` to our previous `xporegress` command:

```
. xporegress react no2_class, controls($cc i.($fc)) selection(cv) rseed(12345)
(output omitted)
Cross-fit partialing-out      Number of obs      =      1,036
linear model                  Number of controls =       32
                             Number of selected controls =    26
                             Number of folds in cross-fit =    10
                             Number of resamples      =       1
                             Wald chi2(1)            =    23.34
                             Prob > chi2             =    0.0000
```

react	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
no2_class	2.348458	.4861133	4.83	0.000	1.395693	3.301222

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

If you run this command, you will see that cross-validation takes much longer than plugin. For each cross-fit, cross-validation performs its own 10-way partition of the data and runs lassos on each of those 10 partitions for the variables `react` and `no2_class`. After all this computation, the results look remarkably similar. Our coefficient estimate is still 2.3 and is still highly significant. Our 95% confidence interval is 1.4 to 3.3. This point estimate and the one obtained by plugin are close and well within each respective confidence interval.

This high degree of similarity is not always the case. Sometimes different methods produce different results.

Given that the results are so similar, you might guess that plugin and cross-validation selected similar controls. A quick glance at the header will dispel that thought. Cross-validation selected 26 controls, far more than the 10 controls selected by plugin. Remember that picking the “right” model is not what these methods are about. As long as the selected controls adequately control for everything necessary to fit the variables of interest, they are doing their job.

For these data and this model, the results simply are not very sensitive to the number of controls selected. This is true over a broad range—at the least from the 10 controls selected by plugin to the 26 controls selected by cross-validation.

Let’s take a quick look at the lassos:

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

Variable	Model	Selection method	No. of selected variables		
			min	median	max
no2_class	linear	cv	9	13	16
react	linear	cv	6	15	19

Even within cross-fits, cross-validation shows a lot more variation than plugin. The number of selected controls from the lassos on `no2_class` ranges from 9 to 16. The lassos for `react` show even more variation, ranging from 6 to 19 selected controls. Where did the 26 controls in the output of `xporegress` come from? It is a count of the union of all controls from any lasso.

Let's peer a bit deeper into the lassos by using `lassoinfo`:

```
. lassoinfo, each
      Estimate: active
      Command: xpo regress
```

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

We see that the lasso penalty parameter λ and the associated number of selected variables varies widely. This is particularly true of the lassos for `react`. It simply does not matter; the estimates for `no2_class`, our covariate of interest, are not affected.

2.4 Fitting via double selection (ds) using cross-validation

Continuing with the example above, we will fit the model using double selection and cross-validation. We recommend this for three reasons.

First, the double-selection method works quite a bit differently from the partialing out done by cross-fit. Instead of working with the lasso results one at a time and then using method of moments to estimate the parameters, double selection takes the union of selected covariates from all lassos and then just does a linear regression of `react` on `no2_class` and that union of selected covariates. The two methods are asymptotically equivalent if both sparsity bounds are met, but in finite samples, they can respond differently to any violation of the conditions required by the inferential lasso estimators. See *Solutions that focus on the true model* in [LASSO] [Lasso inference intro](#) for a discussion of sparsity bounds.

Second, double selection requires only two lassos for our model, making it much easier to explore the lassos.

Third, double selection is much easier to explain. We just did it above in half a sentence.

```
. dsregress react no2_class, controls($cc i.($fc)) selection(cv) rseed(12345)
Estimating lasso for react using cv
Estimating lasso for no2_class using cv
Double-selection linear model      Number of obs      =      1,036
                                   Number of controls =      32
                                   Number of selected controls = 22
                                   Wald chi2(1)           =      24.17
                                   Prob > chi2            =      0.0000
```

react	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
no2_class	2.404191	.4890458	4.92	0.000	1.445679	3.362704

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The coefficient estimate for `no2_class` is now to 2.4, still almost the same as fitting by `xporegress` with plugin selection. The associated confidence interval is 1.4 to 3.4. Our test against 0 was strong and is still strong. This really is a benign dataset for these linear models.

As with cross-validation, with cross-fit the number of selected controls is large—22.

What we are seeing are incredibly stable estimates.

2.5 Fitting via the other 22 methods

We will not show the results of the other 22 methods for fitting this model. Here is what you would type for each method:

```
. xpo regress react no2_class, controls($cc i.($fc)) selection(adaptive) rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) selection(cv) rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) selection(adaptive) rseed(12345)
. dsregress react no2_class, controls($cc i.($fc)) rseed(12345)
. dsregress react no2_class, controls($cc i.($fc)) selection(adaptive) rseed(12345)
. xpo regress react no2_class, controls($cc i.($fc)) sqrtlasso rseed(12345)
. xpo regress react no2_class, controls($cc i.($fc)) selection(cv) sqrtlasso ///
  rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) sqrtlasso rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) selection(cv) sqrtlasso ///
  rseed(12345)
. dsregress react no2_class, controls($cc i.($fc)) sqrtlasso rseed(12345)
. dsregress react no2_class, controls($cc i.($fc)) selection(cv) sqrtlasso ///
  rseed(12345)
. xpo regress react no2_class, controls($cc i.($fc)) semi rseed(12345)
. xpo regress react no2_class, controls($cc i.($fc)) selection(cv) semi rseed(12345)
. xpo regress react no2_class, controls($cc i.($fc)) selection(adaptive) semi ///
  rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) semi rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) selection(cv) semi rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) selection(adaptive) semi ///
  rseed(12345)
. xpo regress react no2_class, controls($cc i.($fc)) sqrtlasso semi rseed(12345)
. xpo regress react no2_class, controls($cc i.($fc)) selection(cv) sqrtlasso semi ///
  rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) sqrtlasso semi rseed(12345)
. poregress react no2_class, controls($cc i.($fc)) selection(cv) sqrtlasso semi ///
  rseed(12345)
```

By now, the commands are nearly self explanatory.

Command `xpo regress` fits via the cross-fit partialing-out method. Command `poregress` fits via the partialing-out method. Command `dsregress` fits via the double-selection method.

Adding option `selection(cv)` specifies that cross-validation select the covariates. Adding option `selection(adaptive)` specifies that adaptive lasso select the covariates. No `selection()` option implies that the plugin method (the default) select the covariates.

Adding option `sqrtlasso` specifies the square-root lasso rather than standard lasso.

Adding option `semi` specifies an alternate way of combining the moments for the `po` and `xpo` methods.

If you are interested, run some or all of these commands.

If you do, you will find that for these data and this model, the method we choose makes little difference. The results for these 22 methods look a lot like the results for the first 3 methods. The maximum coefficient for `no2_class` is 2.4, and the minimum coefficient is 2.3. The maximum standard error is 0.51, and the minimum is 0.48. All methods reject that the coefficient for `no2_class` is 0 well beyond the 1% level of significance.

The close similarity of the results from all 25 methods may seem surprising. Are they all selecting the same controls? The answer is no. Recall from [2.2 Fitting via cross-fit partialing out \(xpo\) using plugin](#) that the selected number of controls is 10, whereas from [2.4 Fitting via double selection \(ds\) using cross-validation](#), the selected number of controls is 22—over twice as many.

Let's look at just two of the methods to see which controls they are selecting. We can easily do this only lasso by lasso (not command by command), so we will use two double-selection methods. Double selection creates only two lassos for our model. Comparing the cross-fit methods would require looking at 20 lassos per method. Let's use `lassocoeff` to compare double selection using plugin and double selection using cross-validation.

First, we rerun those two models and store their estimates.

```
. dsregress react no2_class, controls($cc i.($fc)) rseed(12345)
  (output omitted)
. estimates store ds_plugin
. dsregress react no2_class, controls($cc i.($fc)) selection(cv) rseed(12345)
  (output omitted)
. estimates store ds_cv
```

Then, we compare the selected controls from each lasso.

```
. lassocoef (ds_plugin, for(react))
>           (ds_cv      , for(react))
>           (ds_plugin, for(no2_class))
>           (ds_cv      , for(no2_class))
```

	ds_plugin react	ds_cv react	ds_plugin no2_class	ds_cv no2_class
age	x	x		x
0.sex	x	x		
grade				
2nd	x	x		
4th	x	x		
3rd				x
feducation				
University	x	x		x
Primary		x		
<Primary				x
age0		x		
sev_home		x		x
siblings_young		x		x
0.lbweight		x		
meducation				
1		x		
2		x		
no2_home			x	x
green_home			x	x
noise_school			x	x
sev_school			x	x
precip			x	x
breastfeed				
No breastfeeding				x
>6 months				x
msmoke				
No smoking				x
_cons	x	x	x	x

Legend:

```
b - base level
e - empty cell
o - omitted
x - estimated
```

The first two columns of x's show which controls were selected from the lassos for the dependent variable, `react`—the first column for the plugin method and the second for cross-validation. The third and fourth columns of x's show which controls were selected by the lassos for the covariate of interest, `no2_class`.

Cross-validation selected more controls than did plugin in the lassos for both the dependent variable, `react`, and the covariate of interest, `no2_class`. That is not surprising because plugin is designed to be cautious about adding noise through variable selection while cross-validation cares only about minimizing the cross-validation mean squared error.

Perhaps more interesting is that for both `react` and `no2_class`, cross-validation selected a superset of the variables selected by plugin. While not guaranteed, that result is a reflection of how the lasso works. Plugin and cross-validation select their covariates by setting an “optimal” value of λ , the lasso penalty. Plugin selects a larger λ and thereby a stronger penalty that selects fewer variables. As the penalty gets weaker, lasso can drop selected variables when adding others, but lasso is more likely to simply add variables. So, in this case, cross-validation’s weaker penalty leads to a superset of the variables selected by plugin. That is a bit of an oversimplification because plugin selects variables that have been weighted by the inverse standard deviation of their scores while cross-validation does not weight the variables. This means that the lambda for plugin and the lambda for cross-validation are on different scales.

Recall, though, that the only role of the selected controls is to adequately capture the unmodeled correlations among the dependent variable, the variables of interest, and the model’s error.

2.6 Fitting models with several variables of interest

All 11 inferential models in Stata allow you to have more than one variable of interest. Let’s extend our base example from section 2.2 to include both `no2_class` and `student’s age` as variables of interest.

The only trick is that we must remove our new variables of interest from our list of continuous controls. `vl` makes that easy:

```
. vl create cc6 = cc - (age)
note: $cc6 initialized with 9 variables.
```

We have now created the global macro `cc6`, which has the same variables as `cc` except that `age` has been removed.

We fit the model using cross-fit partialing-out with the default plugin selection by typing

```
. xporegress react no2_class age, controls($cc6 i.($fc)) rseed(12345)
(output omitted)
```

Cross-fit partialing-out	Number of obs	=	1,036
linear model	Number of controls	=	31
	Number of selected controls	=	9
	Number of folds in cross-fit	=	10
	Number of resamples	=	1
	Wald chi2(2)	=	25.24
	Prob > chi2	=	0.0000

react	Robust		z	P> z	[95% conf. interval]	
	Coefficient	std. err.				
no2_class	2.353826	.4892462	4.81	0.000	1.394921	3.312731
age	-25.01451	11.38901	-2.20	0.028	-47.33656	-2.69245

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The coefficient for `no2_class` has barely changed at all from its estimate in section 2.2.

Again, we interpret the coefficients on the variables of interest just as we would if they were part of a standard linear regression. So a 1-unit change in `no2_class` elicits a 2.4-unit change in `react`. A 1-unit change in `age` elicits a -25-unit change in `react`. Because the relationship is linear, these changes can be interpreted as the expected change for any individual or as the expected change for any population or subpopulation of interest.

2.7 Fitting models with factor variables of interest

Having a factor variable of interest is really no different than having several variables of interest. Factor variables just provide a convenient way to add several indicator variables to our model.

Those who study response times for children know that they decrease (improve) as the child is exposed over time to educational stimuli. We might then be interested in how the response times vary across the child's grade level. Ignoring our original interest in the effect of nitrogen dioxide for the moment, let's pretend our only variable of interest is `grade` in school.

The distribution of grades in our sample looks like this:

```
. tabulate grade
```

Grade in school	Freq.	Percent	Cum.
2nd	412	37.83	37.83
3rd	397	36.46	74.29
4th	280	25.71	100.00
Total	1,089	100.00	

If we wish to use the levels of `grade` as our variables of interest, we need to remove it from our list of factor-variable controls:

```
. vl create fc7 = fc - (grade)
note: $fc7 initialized with 7 variables.
```

We are not currently interested in the effect of nitrogen dioxide, so we need to add it back to the list of continuous controls:

```
. vl create cc7 = cc + (no2_class)
note: $cc7 initialized with 11 variables.
```

We can now fit our model with the levels of `grade` as our variables of interest. We add the option `baselevels` so that we can see which level of grade has been made the base level.

```
. xppregress react i.grade, controls($cc7 i.($fc7)) baselevels rseed(12345)
(output omitted)
```

```
Cross-fit partialing-out          Number of obs          =          1,036
linear model                      Number of controls      =           30
                                  Number of selected controls =           5
                                  Number of folds in cross-fit =          10
                                  Number of resamples          =           1
                                  Wald chi2(2)                  =          16.82
                                  Prob > chi2                    =           0.0002
```

react	Robust		z	P> z	[95% conf. interval]	
	Coefficient	std. err.				
grade						
2nd	0	(base)				
3rd	-62.07497	15.26513	-4.07	0.000	-91.99408	-32.15587
4th	-92.52593	25.02151	-3.70	0.000	-141.5672	-43.48467

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

A common theme in these sections is that we interpret the results for the variables of interest just as we would if they were part of a linear regression. This is no different for factor variables. As we would with a simple linear regression, we interpret each of the coefficients as the increases in performance relative to the base level for second graders. We can see that mean reaction time is 62 milliseconds faster for third graders than it is for second graders. Fourth graders are, on average, 93 milliseconds faster than second graders.

That common theme extends to the tools that are available after fitting a model with any of the lasso inference commands. For example, we can use `contrast` to do comparisons of the grade levels that are not against a reference category, as they were in the regression. We could use a reverse adjacent (`ar.`) contrast to compare each grade to the prior grade:

```
. contrast ar.grade
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
grade			
(3rd vs 2nd)	1	16.54	0.0000
(4th vs 3rd)	1	4.23	0.0397
Joint	2	16.82	0.0002

	Contrast	Std. err.	[95% conf. interval]	
grade				
(3rd vs 2nd)	-62.07497	15.26513	-91.99408	-32.15587
(4th vs 3rd)	-30.45096	14.80702	-59.47218	-1.429727

The regression showed a 62-millisecond decrease in response time when comparing third graders to second graders, and that is reproduced by `contrast`. The difference with reverse-adjacent comparisons is that the comparison group for fourth graders is now third graders, and we estimate that difference to be a 30-millisecond decrease. It would take a bit more work to determine if the apparently slower improvement from third to fourth grade is indeed significantly different from the improvement from second to third grade. If you are interested, and without explanation, you could type

```
. contrast ar.grade, post
. lincom _b[ar2vs1.grade] - _b[ar3vs2.grade]
```

You will find that, by a slim margin, we fail to distinguish between the effect of going from second to third grade and the effect of going from third to fourth grade.

If we had a more complicated set of interest, we would find `contrast` indispensable. If you have factor variables of interest, we suggest you become familiar with `contrast`.

What we cannot do with results from the inferential lasso models is use `margins` to estimate population and subpopulation means. `margins` requires a full coefficient vector and variance matrix for those coefficients. The lasso inference commands can only tell us about a subset of that coefficient vector and associated variance matrix.

If you are epidemiologically inclined, you might wonder if the effect of `grade` is not just a proxy for increasing age. Now that we have multiple variables of interest and factor variables of interest, we can check that too:

```
. vl create cc7b = cc7 - (age)
note: $cc7b initialized with 10 variables.
. xporegress react age i.grade, controls($cc7b i.($fc7)) baselevels rseed(12345)
(output omitted)
```

```
Cross-fit partialing-out      Number of obs      =      1,036
linear model                  Number of controls  =       29
                              Number of selected controls =        3
                              Number of folds in cross-fit =       10
                              Number of resamples      =        1
                              Wald chi2(3)            =      203.93
                              Prob > chi2             =       0.0000
```

react	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
age	-18.50751	11.16037	-1.66	0.097	-40.38143	3.366418
grade	0 (base)					
2nd						
3rd	-67.35294	15.4679	-4.35	0.000	-97.66947	-37.03641
4th	-100.7346	25.0814	-4.02	0.000	-149.8932	-51.57594

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The estimates for grade level have changed only a bit. Response times may improve with age, but we cannot detect that at the 5% level. Regardless, the effect of educational stimulation appears to be independent. Most importantly, we see that all of our contrast tools can be used with these estimators.

2.8 Fitting models with interactions of interest

Not surprisingly, tools for evaluating interactions for other estimation commands are also available to evaluate interactions among our variables of interest, whether those interactions are strictly among factor variables or are with factor variables and continuous variables. Let's arbitrarily check for an interaction between the child's sex and his or her age. Again, we need to manage our list of controls by removing `sex` and `age` from the list of factor-variable controls. And we again need to put `no2_class`, which is no longer a variable of interest, back into the continuous controls.

```
. vl create fc8 = fc - (sex grade)
note: $fc8 initialized with 6 variables.
. vl create cc8 = cc + (no2_class)
note: $cc8 initialized with 11 variables.
```

We can then fit a cross-fit model of reaction time where our variable of interest is `sex##grade`—the interaction of `sex` and `grade` while also including individual indicators for the levels of `sex` and `grade`.

```
. xppregress react sex##grade, controls($cc8 i.($fc8)) baselevels rseed(12345)
(output omitted)
Cross-fit partialing-out      Number of obs      =      1,036
linear model                  Number of controls  =         28
                              Number of selected controls =         6
                              Number of folds in cross-fit =        10
                              Number of resamples      =         1
                              Wald chi2(5)             =        64.57
                              Prob > chi2            =         0.0000
```

react	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
sex						
Male	0	(base)				
Female	45.10077	13.73912	3.28	0.001	18.17259	72.02896
grade						
2nd	0	(base)				
3rd	-65.62381	17.72386	-3.70	0.000	-100.362	-30.88568
4th	-102.2437	26.5379	-3.85	0.000	-154.257	-50.23033
sex#grade						
Female#3rd	3.173242	19.09434	0.17	0.868	-34.25098	40.59747
Female#4th	18.42495	19.98327	0.92	0.357	-20.74154	57.59144

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The two coefficients of the interaction `sex#grade` and their associated statistics do not give us much hope that an interaction is statistically detectable. Let's check anyway:

```
. contrast sex#grade
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
sex#grade	2	0.96	0.6188

Definitely not statistically significant, at any level.

What about the individual effects of `sex` and `grade`?

```
. contrast sex
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
sex	1	42.33	0.0000

```
. contrast grade
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
grade	2	17.83	0.0001

Both individual effects are significant at any level you would care to consider.

Some studies have found differences in some types of reaction times between the sexes, but we might want to consider another factor—the interaction between `sex` and `no2_class`.

We can put `grade` back into the controls because it has no interaction with `sex`.

```
. vl create fc8b = fc - (sex)
note: $fc8b initialized with 7 variables.
```

We are ready to fit a model that includes `sex`, `no2_class`, and their interaction. That can be written in shorthand, by typing `c.no2_class##i.sex`. We fit the model:

```
. xporegress react c.no2_class##i.sex, controls($cc i.($fc8b)) rseed(12345)
(output omitted)
```

```
Cross-fit partialing-out          Number of obs          =          1,036
linear model                      Number of controls     =             30
                                Number of selected controls =             9
                                Number of folds in cross-fit =            10
                                Number of resamples         =             1
                                Wald chi2(3)                =            63.42
                                Prob > chi2                 =            0.0000
```

react	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
no2_class	1.708798	.5961435	2.87	0.004	.5403779	2.877217
sex						
Female	17.47061	24.31548	0.72	0.472	-30.18686	65.12807
sex#						
c.no2_class						
Female	1.099669	.7737183	1.42	0.155	-.4167913	2.616129

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

Everything we need to know is in this output.

The effect of `no2_class` is still positive, as it was for all of our earlier fits. The effect is now a bit smaller at a 1.7-millisecond increase in response for every microgram increase in NO_2 per cubic meter.

There is no longer a significant difference in response times for females compared with males. The point estimate is 17, but its z statistic is a scant 0.72.

The interaction between `sex` and `no2_class` is also not significant, though you might wish you had more data.

You might be curious if the effect of nitrogen dioxide across both males and females from this model is similar to our earlier models without an interaction. If we assume 50% males and 50% females, we just need to add half of the interaction term to the estimate for males.

```
. lincom no2_class + .5*c.no2_class#1.sex
(1) no2_class + .5*1.sex#c.no2_class = 0
```

react	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
(1)	2.258632	.4800889	4.70	0.000	1.317675	3.199589

The estimate is extremely close to the point estimate and standard errors that we obtained in [2.2 Fitting via cross-fit partialing out \(xpo\) using plugin](#)—both round to 2.3 with standard errors that round to 0.48.

While we have pretended to be performing analysis, the important thing to know is that the standard inference tools can be applied to the variables of interest.

2.9 Fitting models with a nonlinear relationship of interest

Let’s continue with our reaction-time example and put a nonlinearity in `no2_class` into the covariates of interest. What we really mean by “nonlinear” in this context is nonlinear-but-linearizable—polynomials, logs, ratios, and the like.

We just want to demonstrate how to think about nonlinearities with these models, so let’s not dwell on where the nonlinear relationship comes from. In your work, you may have some theory or precedence for your choice of nonlinearities. For now, we know that fractional polynomials (`fp`) produce whole classes of reasonable curves, so we will arbitrarily pick one of those forms that allows for two inflection points—including one over the square root and the cube of the variable.

```
. generate no2fp1 = no2_class^(-2)
. generate no2fp2 = no2_class^3
```

With those as our two covariates of interest, we fit a cross-fit model. Our controls are from the model we fit in [2.2 Fitting via cross-fit partialing out \(xpo\) using plugin](#).

```
. xporegress react no2fp1 no2fp2, controls($cc i.($fc)) rseed(12345)
(output omitted)
```

```
Cross-fit partialing-out      Number of obs      =      1,036
linear model                  Number of controls  =       32
                              Number of selected controls =       11
                              Number of folds in cross-fit =       10
                              Number of resamples         =        1
                              Wald chi2(2)                =      24.55
                              Prob > chi2                 =      0.0000
```

react	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
no2fp1	-2915.067	2227.731	-1.31	0.191	-7281.339	1451.205
no2fp2	.0005923	.0001394	4.25	0.000	.0003191	.0008655

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

We see that it is unclear if we really need two terms to model this relationship. Only one of the terms is significant. But our nonlinearity is just a construct for demonstration and we want to see how this works, so we are undeterred.

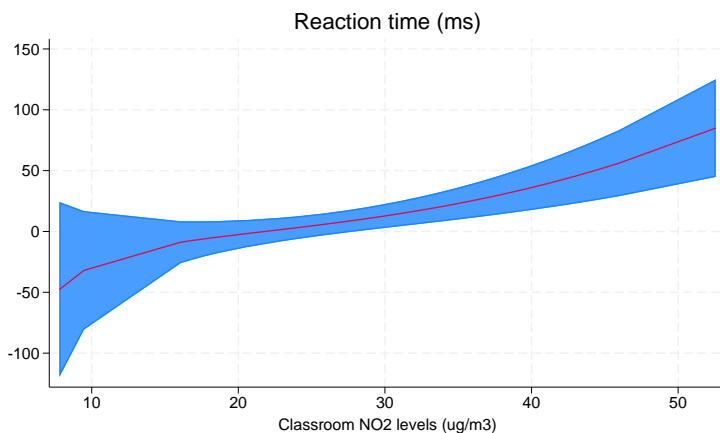
We could do a bit of algebra and decide what those terms imply about the relationship between nitrogen dioxide and reaction time. Or we could just graph the relationship. Predictions in inferential models are typically not much use, but they are perfect for our purpose.

We say predictions are not much use because the selected controls should not be used in a prediction. They are used by `xporegress` solely to obtain consistent estimates of the model of interest, but they are not themselves interpretable. So they should not be used to form predictions. We should not even use the intercept. For `xporegress` and all the other inferential models, only our covariates of interest affect the prediction. That is fine with us; that is all we want to see. We would like to get confidence intervals too, so let's use `predictnl` to get our predictions:

```
. predictnl reacthat = predict(), ci(1b ub)
note: confidence intervals calculated using Z critical values.
```

We can then graph the prediction and its confidence interval:

```
. twoway rarea lb ub no2_class, sort || line reacthat no2_class,
> sort legend(off) title("Reaction time (ms)")
```



There might be some upward curvature as nitrogen dioxide reaches its highest observed levels, but the confidence interval is too wide to be sure. The downward bend at the lowest levels of nitrogen dioxide is also suspect because the confidence interval is also wide in that region of the graph. We have scant evidence that this curve is any better than a linear relationship.

If you are unfamiliar with `twoway` syntax, we asked for two overlaid plots: a range area for the confidence interval from the variables `lb` and `ub` plotted against `no2_class`, `rarea lb ub no2_class`, and a line of predicted reaction time from the variable `reacthat` against `no2_class`.

Unfortunately, we cannot use any information-criterion tools to compare our nonlinear fit with our earlier linear fit. The inferential models cannot estimate the log likelihood or any form of residual required to form any information-criterion measures.

2.10 Controls are controls

The literature on the inferential models fit by double-selection, partialing-out, and cross-fit partialing-out estimators refers to the “variables of interest”, but a more accurate term might be “submodel of interest”. We say that because a maintained assumption is that the control variables are just controls and they do not interact with the variable or variables of interest. That is to say, they can shift the expected value of the outcome, but they cannot change the effect of the variables of interest.

If you think control variable `x3` actually interacts with one of your variables of interest, say, `d1`, then you will need to include that interaction in your submodel of interest. So if `x3` and `d1` are continuous, you need to add `c.x3#c.d1` to your submodel of interest; if `x3` is an indicator or multi-value factor variable, you need to add `i.x3#c.d1`; if both are factor variables, you need to add `i.x3#i.d1`. In these cases, `x3` is not a control variable—it is part of your submodel of interest.

3 Fitting logit inferential models to binary outcomes. What is different?

Even if your current interest is logit models, we suggest you also read [2 Fitting and interpreting inferential models](#). That section has many more examples and goes into more detail. If you are starting here, we also suggest you read [1.4 The primary dataset](#) to become familiar with the dataset and how we are manipulating it. Section 1.4 is not essential reading, but if things become confusing, do read it. Here we focus primarily on what is different about logit models.

Without exception, every command and example from section 2 can be run using a logit lasso inference command. Just change `regress` to `logit` in the estimation commands, and change the dependent variable from `react` to the dependent variable we create below.

We will replicate a few of the analyses from section 2 using logit models and explain how the results are interpreted with binary outcomes. Feel free to run others. Their results are interpreted in the same way as those shown here.

Let’s continue with the [dataset](#) we have been using to measure the effect of nitrogen dioxide in the classroom on the reaction time of school children.

```
. use https://www.stata-press.com/data/r18/breathe, clear
(Nitrogen dioxide and attention)
```

We need to create the global macros that will hold our lists of continuous and factor-variable control variables:

```
. do https://www.stata-press.com/data/r18/no2
(output omitted)
```

To see how these lists were created, see [1.4 The primary dataset](#).

This dataset does not have a binary (dichotomous) dependent variable, but it is easy enough to create one. The variable `omissions` contains a count of the number of times a child failed to respond to a stimuli. We can pretend that we only saw whether or not there were any omissions. Let’s create a variable that is 1 when there were any omissions and is 0 otherwise:

```
. generate miss1 = omissions >= 1 if !missing(omissions)
(5 missing values generated)
```

Then take a quick look at our new variable:

```
. tabulate miss1
```

miss1	Freq.	Percent	Cum.
0	508	46.86	46.86
1	576	53.14	100.00
Total	1,084	100.00	

We have 508 children who never missed a stimulus from the test and 576 who missed at least one stimulus.

3.1 Interpreting standard odds ratios

If you are new to inferential lasso models and have not at least read [2.2 Fitting via cross-fit partialing out \(xpo\) using plugin](#), do that now. We will only explain how to interpret the odds ratios below. Section 2.2 explains more.

We can now fit a model of how classroom nitrogen dioxide levels (`no2_class`) affect whether children miss any stimuli on a reaction-time test (`miss1`). Our continuous controls are in the [global macro \\$cc](#) and our [factor-variable](#) controls are in the global macro `$fc`, as they were in our very first example in section 2.2. We use `xpologit` to fit the model:

```
. xpologit miss1 no2_class, controls($cc i.($fc)) rseed(12345)
(output omitted)
Cross-fit partialing-out      Number of obs      =      1,036
logit model                  Number of controls  =         32
                             Number of selected controls =         5
                             Number of folds in cross-fit  =        10
                             Number of resamples          =         1
                             Wald chi2(1)                =       11.18
                             Prob > chi2                 =       0.0008
```

miss1	Odds ratio	Robust std. err.	z	P> z	[95% conf. interval]
no2_class	1.027338	.0082854	3.34	0.001	1.011227 1.043706

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The odds ratio for `no2_class` is 1.03. We interpret that ratio just as we would if this were a [logistic](#) regression. For every unit increase in the level of nitrogen dioxide, the odds of a student missing at least one stimulus increase by a factor of 1.03, with a confidence interval of 1.01 to 1.04. As always with these models, we cannot estimate a constant, so we do not know the baseline odds.

At face value, that is a small odds ratio, but the range of `no2_class` is 7.8 to 52.6:

```
. summarize no2_class
```

Variable	Obs	Mean	Std. dev.	Min	Max
no2_class	1,089	30.16779	9.895886	7.794096	52.56397

The difference is over 44 micrograms per cubic meter. What odds ratio do we obtain if we increase nitrogen dioxide levels by 44?

```
. lincom _b[no2_class]*44, or
( 1) 44*no2_class = 0
```

miss1	Odds ratio	Std. err.	z	P> z	[95% conf. interval]	
(1)	3.276333	1.162629	3.34	0.001	1.634306	6.568144

The odds go up by 3.3 with a confidence interval from 1.6 to 6.6.

Be careful if you do this by hand. The `or` option did quite a bit of work. There are several ways to write what `lincom` did behind the scenes. One way is

$$OR_{44} = 1.027338^{44}$$

This follows directly from what we said about the original odds ratio being the factor by which odds increase.

Equivalently, and what really happens behind the scenes, is

$$OR_{44} = e^{\beta * 44}$$

where β is the coefficient on `no2_class`, which is the log of the odds ratio shown on the `xpologit` results. These expressions produce identical results.

We said earlier that `xpologit` cannot estimate a baseline odds. It cannot estimate any odds, only odds ratios. Even so, we might consider the degree of these effects by looking at children experiencing truly low nitrogen dioxide levels, say, below 10:

```
. table miss1 if no2_class < 10
```

	Frequency
miss1	
0	24
1	10
Total	34

That gives an odds of $10/24 = 0.42$, or roughly one child missing a stimulus for every two who respond to every stimulus. If we assume that is the starting odds for a child and then increase the nitrogen dioxide levels by 44, the odds move all the way to $3.2 \times 0.42 = 1.3$. At that level of nitrogen dioxide, almost three children miss at least one stimulus for every two who respond to every stimulus.

3.2 Interpreting models with factor variables, nonlinear relationships, and interactions

Let's run through most of the examples that we first demonstrated with linear regression. We are going to set the models up quickly. Read sections 2.6 through 2.9 for more about the models. We will use the same tools; we will just ask them to report odds ratios.

In *2.6 Fitting models with several variables of interest*, we added `age` to our covariates of interest. That means we must pull `age` from our list of continuous controls.

```
. v1 create cc31 = cc - (age)
note: $cc31 initialized with 9 variables.
```

We will use different global macro names throughout this section to avoid collisions with the original examples. These globals hold the same variable lists—they just have a different name.

We fit the model:

```
. xpologit miss1 no2_class age, controls($cc31 i.($fc)) rseed(12345)
(output omitted)
Cross-fit partialing-out          Number of obs          =          1,036
logit model                       Number of controls     =             31
                                   Number of selected controls =             7
                                   Number of folds in cross-fit =            10
                                   Number of resamples      =             1
                                   Wald chi2(2)              =            13.58
                                   Prob > chi2              =            0.0011
```

miss1	Odds ratio	Robust std. err.	z	P> z	[95% conf. interval]	
no2_class	1.048213	.0760006	0.65	0.516	.9093542	1.208275
age	.7922585	.0647357	-2.85	0.004	.6750174	.9298628

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

There is not much to say about the results. Interpret the odds ratios as you would any logistic model with two covariates. The odds ratio for age is 0.79 and is significant at the 5% level with a 95% confidence interval from 0.68 to 0.93. So, older children are less likely to miss a stimulus. We also note that `no2_class` is now insignificant. We are asking a lot of a binary outcome signal.

In [2.7 Fitting models with factor variables of interest](#), we decided that we were interested in the effect of the grade the child was in at school and no longer interested in nitrogen dioxide.

We will set our controls to reflect this:

```
. vl create fc32 = fc - (grade)
note: $fc32 initialized with 7 variables.
. vl create cc32 = cc + (no2_class)
note: $cc32 initialized with 11 variables.
```

And we fit the `xpologit` model:

```
. xpologit miss1 i.grade, controls($cc32 i.($fc32)) baselevels rseed(12345)
(output omitted)
Cross-fit partialing-out      Number of obs      =      1,036
logit model                   Number of controls  =         30
                              Number of selected controls =         3
                              Number of folds in cross-fit =        10
                              Number of resamples      =         1
                              Wald chi2(2)             =         5.51
                              Prob > chi2             =         0.0637
```

miss1	Odds ratio	Robust std. err.	z	P> z	[95% conf. interval]	
grade						
2nd	1	(base)				
3rd	.6371055	.1232829	-2.33	0.020	.4360134	.9309425
4th	.6156729	.1974266	-1.51	0.130	.3283953	1.154259

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The odds ratio of going from second grade, the base level, to third grade is 0.64 and is significant. The odds ratio of going from second grade to fourth grade is 0.62 and is not statistically significant at the 5% level.

These results are weaker than those for the linear model for reaction time. Even so, we forge on and use `contrast` to look at the grade-to-grade odds ratios. `contrast` knows how to exponentiate results to get odds ratios, but it is not quite as smart as `lincom`. We will need to tell `contrast` to use exponential form (`eform()`) and to label the results as “Odds ratio”:

```
. contrast ar.grade, eform(Odds ratio)
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2	
grade				
(3rd vs 2nd)	1	5.43	0.0198	
(4th vs 3rd)	1	0.02	0.8777	
Joint	2	5.51	0.0637	

	Odds ratio	Std. err.	[95% conf. interval]	
grade				
(3rd vs 2nd)	.6371055	.1232829	.4360134	.9309425
(4th vs 3rd)	.9663594	.2149063	.6249454	1.494291

The first comparison is still between second and third grade. We already discussed that comparison when considering the output from `xpologit`. `contrast` reports the same odds ratio and the same p -value. The second comparison is now between third and fourth grade. The point estimate is an odds ratio of 0.97, almost 1, and it is not a significant ratio at the 5% level.

We will skip section [2.8 Fitting models with interactions of interest](#) because it does not offer any new tools for analyzing odds ratios. You can run that model as an inferential lasso logit model on `miss1`. Just remember to add option `eform(Odds ratio)` to any of the `contrast` commands.

In [2.9 Fitting models with a nonlinear relationship of interest](#), we analyzed a nonlinear relationship between reaction time and nitrogen dioxide levels. Recall from section 2.9 that we arbitrarily chose a nonlinear representation for `no2_class` that allows for two inflection points—one over the square root of `no2_class` and one over the cube of the `no2_class`. If you have already worked through section 2.9 with your current dataset, you already have the two variables for the nonlinearity in your dataset. If not, we will need to create them.

```
. generate no2fp1 = no2_class^(-2)
. generate no2fp2 = no2_class^3
```

With these variables in place, we can fit our nonlinear relationship between `miss1` and `no2_class`.

```
. xpologit miss1 no2fp1 no2fp2, controls($cc i.($fc)) rseed(12345)
(output omitted)
convergence not achieved
      gmm step failed to converge
r(498);
```

That did not end well. Generalized method of moments (GMM) is how `pologit` and `xpologit` combine the scores from the partialing-out process to obtain the parameter estimates for the coefficients of interest. With these data and model, GMM simply could not converge. This happens. In the other examples in this section, we have mentioned that the estimates are not as significant as they were for the linear models on reaction time from section 2. Our binary outcome variable, `miss1`, has much less information than the continuous reaction time variable.

Do we think all is lost? This is the first example of instability, so let's try a little harder. We will warn you that you can try `pologit`, but it fails with the same error.

Let's take the advice from [\[LASSO\] Inference requirements](#) and try cross-validation as our selection technique. We return to the cross-fit estimator:

```
. xpologit miss1 no2fp1 no2fp2, controls($cc i.($fc)) selection(cv) rseed(12345)
(output omitted)
convergence not achieved
      gmm step failed to converge
r(498);
```

This is tough. We cannot even try the alternate suggestion from [\[LASSO\] Inference requirements](#) because we already said that `pologit` with plugin selection failed. We will tell you now that `pologit` with cross-validation selection also fails.

We did say earlier that double selection is more stable. Let's try `dslogit`, first with cross-validation, and store the results:

```
. dslogit miss1 no2fp1 no2fp2, controls($cc i.($fc)) selection(cv) coef
> rseed(12345)
Estimating lasso for miss1 using cv
Estimating lasso for no2fp1 using cv
Estimating lasso for no2fp2 using cv
Double-selection logit model      Number of obs      =      1,036
                                Number of controls  =          32
                                Number of selected controls =         23
                                Wald chi2(2)              =         16.19
                                Prob > chi2              =         0.0003
```

miss1	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
no2fp1	-79.45294	41.32577	-1.92	0.055	-160.45	1.544089
no2fp2	7.18e-06	2.52e-06	2.85	0.004	2.24e-06	.0000121

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

```
. estimates store ds_cv
```

We have estimates. There is nothing to suggest instability in these results. The coefficient on `no2fp2` is tiny, but that is the cube of `no2_class`. It needs to be a small coefficient.

What does the plugin selection method have to say?

```
. dslogit miss1 no2fp1 no2fp2, controls($cc i.($fc)) coef rseed(12345)
Estimating lasso for miss1 using plugin
Estimating lasso for no2fp1 using plugin
Estimating lasso for no2fp2 using plugin
Double-selection logit model      Number of obs      =      1,036
                                Number of controls  =          32
                                Number of selected controls =          5
                                Wald chi2(2)              =         14.63
                                Prob > chi2              =         0.0007
```

miss1	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
no2fp1	-80.76289	39.2933	-2.06	0.040	-157.7763	-3.749442
no2fp2	6.01e-06	2.35e-06	2.56	0.010	1.41e-06	.0000106

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

```
. estimates store ds_plugin
```

Those coefficients look similar to the ones from cross-validation selection. What is more, plugin selected only 5 controls whereas cross-validation selected 23. The double-selection results are similar over a wide range of selected controls. We stored the results from the estimators, so let's peek at the controls from the two methods by using `[LASSO] lassoinfo`:

```
. lassoinfo ds_cv ds_plugin
```

```
Estimate: ds_cv
Command: dslogit
```

Variable	Model	Selection method	Selection criterion	lambda	No. of selected variables
miss1	logit	cv	CV min.	.0229644	6
no2fp1	linear	cv	CV min.	.0000249	17
no2fp2	linear	cv	CV min.	636.8366	13

```
Estimate: ds_plugin
Command: dslogit
```

Variable	Model	Selection method	lambda	No. of selected variables
miss1	logit	plugin	.07161	0
no2fp1	linear	plugin	.1199154	4
no2fp2	linear	plugin	.1199154	4

Cross-validation is selecting many more controls for each variable's lasso: for `miss1`, 6 versus 0; for `no2fp1`, 17 versus 4; and for `no2fp2`, 13 versus 4.

Let's look more closely with `lassocoef`:

```
. lassocoef (ds_plugin, for(miss1))
>           (ds_cv      , for(miss1))
>           (ds_plugin, for(no2fp1))
>           (ds_cv      , for(no2fp1))
>           (ds_plugin, for(no2fp2))
>           (ds_cv      , for(no2fp2))
```

	ds_plugin miss1	ds_cv miss1	ds_plugin no2fp1	ds_cv no2fp1	ds_plugin no2fp2	ds_cv no2fp2
age		x				x
grade						
2nd		x				
4th				x		
3rd						x
0.overweight		x		x		
feducation						
University		x				x
<Primary				x		x
Primary				x		
no2fp1		x				
no2fp2		x				
no2_home			x	x		x
green_home			x	x	x	x
noise_school			x	x	x	x
precip			x	x	x	x
age0				x		
sev_home				x		x
sev_school				x	x	x
siblings_old				x		
0.sex				x		
breastfeed						
<6 months				x		
>6 months				x		x
meducation						
1				x		
2				x		
msmoke						
No smoking						x
Smoking						x
_cons	x	x	x	x	x	x

Legend:

```
b - base level
e - empty cell
o - omitted
x - estimated
```

A careful perusal of the x's shows that cross-validation selected each control that plugin selected for all lassos. It also selected many more controls. We have seen this behavior [before](#). At least we are not worried that the selection method produces different results.

We graphed the nonlinear effect of nitrogen dioxide on reaction time by using a linear model in section 2.9. The path of coefficients from a logit model do not have any interpretation. Wait! The results that we saw at the beginning of this section were interpretable. All we had to do was exponentiate the total difference from some baseline and we obtained an odds ratio. We can do that here too.

The `predict` command will give us the linear predictions for just the two fractional polynomial terms. We want a confidence interval (CI), so let's use `predictnl`:

```
. predictnl xbhat = predict(), ci(xblb xub)
note: confidence intervals calculated using Z critical values.
```

We have no intercept, so we need to pick a level of `xbhat` whose exponential will be our baseline odds. Do we think the minimum value of nitrogen dioxide is reasonable? Or do we think that is an outlier?

```
. summarize no2_class, detail
```

Classroom NO2 levels (ug/m3)					
	Percentiles	Smallest			
1%	9.474087	7.794096			
5%	16.86244	7.794096			
10%	18.5384	7.794096	Obs	1,089	
25%	22.81843	7.794096	Sum of wgt.	1,089	
50%	29.91033		Mean	30.16779	
		Largest	Std. dev.	9.895886	
75%	36.59826	52.56397			
90%	42.04398	52.56397	Variance	97.92857	
95%	45.97548	52.56397	Skewness	.2082405	
99%	52.52346	52.56397	Kurtosis	2.63782	

We have five identical values of 7.8 for at least the smallest five, and they are not far from the first percentile. If we can find the linear prediction for the minimum value of `no2_class`, that would be a serviceable baseline.

```
. summarize xbhat if no2_class <= r(min)
```

Variable	Obs	Mean	Std. dev.	Min	Max
xbhat	8	-1.326629	0	-1.326629	-1.326629

The `r(min)` in that expression was just a saved result from the previous `summarize` command. It contained the minimum of `no2_class`. Our linear prediction that corresponds to the minimum of `no2_class` is -1.3 . That is our linear baseline. We could subtract -1.3 from our linear prediction and its bounds, but the value is stored in higher precision in `r(mean)`. Let's subtract our baseline and exponentiate the results to obtain odds ratios:

```
. generate orhat = exp(xbhat - r(mean))
. generate orlb = exp(xblb - r(mean))
. generate orub = exp(xub - r(mean))
```

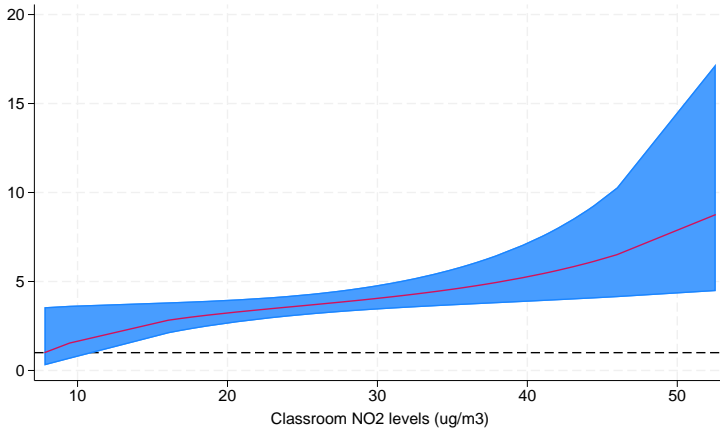
Let's label the variable holding our point estimate of the odds ratios.

```
. label variable orhat "Odds ratio vs. lowest levels of NO2"
```

It is always good to label your variables. And we would like a little labeling on our graph. If you have lost track of what we are computing, that label should be a hint.

That was a bit of work. And, admittedly, it was only loosely tied to the algebra at the top of this section. Was it worth it? What do we have?

```
. twoway rarea orlb orub no2_class, sort || line orhat no2_class,
> yline(1) legend(off) sort
```



Well, it is pretty, in a statistical way. The lowest value of the red line is exactly 1.0. It is the baseline odds that we assigned to the lowest levels of `no2_class`. We did that when we subtracted the prediction for the lowest levels of `no2_class` from all of our predictions. That made the lowest prediction exactly 0 and its exponential 1.0—meaning no effect. That was done by construction.

Let's look at the other end of the graph, the rightmost portion where `no2_class` levels are just above 50. The red line now looks to be between 8 and 9—we will just say 8. The odds of a child missing a stimuli when nitrogen dioxide levels are above 50 are 8 times higher than the odds when nitrogen dioxide levels are at the minimum in the dataset. For nitrogen dioxide levels of 30, the red odds-ratio line looks to be about 4, meaning that children facing levels of 30 have 4 times higher odds of missing a stimuli than do children facing the lowest levels of nitrogen dioxide. And so on. The line traces out the odds ratio for each level of nitrogen dioxide against the odds for the lowest level of nitrogen dioxide.

The blue area is the 95% confidence boundary for the odds ratio. The boundary is pretty narrow for the majority of the curve, but it expands as nitrogen dioxide levels exceed 35 or 40. At the highest levels, the band ranges from about 4 all the way to about 17.

We drew a black reference line at 1.0 because an odds ratio of 1.0 means no effect. At the lowest levels of nitrogen dioxide, the lower bound of the CI is below 1.0. So at those levels, we cannot tell whether nitrogen dioxide has an effect.

The point estimates and their CIs are in the variables `orhat`, `orlb`, and `orub`. You can summarize them or look at them for specific levels of `no2_class`.

Making the lowest level of `no2_class` the reference odds was arbitrary. Rather than subtract the mean of the linear prediction for that level of `no2_class`, we could have used the value at the mean of `no2_class`, or the median, or any value we choose. We need not have considered `no2_class` at all in setting the baseline. Any of these changes would just shift the curves up or down. Their relative positions do not change. If you have a specific comparison in mind, change the baseline.

All that said, the CIs are wide and we might be curious whether a straight line fits just as well. As we mentioned in section 2.9, the standard AIC and BIC methods for choosing among specifications are not possible after inferential lasso estimation. We are pretty much stuck with eyeing it. If you want to do that, do not try with this graph. The exponential has put its own curve onto the odds ratios. Look instead at a graph of the original predictions:

```
twoway rarea xblb xbsub no2_class, sort || line xbhat no2_class, sort
```

We leave you to draw that yourself.

4 Fitting inferential models to count outcomes. What is different?

Even if your current interest is Poisson models, we suggest you also read [2 Fitting and interpreting inferential models](#). That section has many more examples and goes into more detail. If you are starting here, we also suggest you read [1.4 The primary dataset](#) to become familiar with the dataset and how we are manipulating it. Section 1.4 is not essential reading, but it does explain more about how we manage the variable lists in this entry. Here we focus primarily on what is different about Poisson models.

Every command and example from section 2 can be run using a Poisson lasso inference command. Just change `regress` to `poisson` in the estimation commands, and change the dependent variable from `react` to `omissions`.

We will replicate a few of the analyses from section 2 using Poisson models and explain how the results are interpreted with count outcomes. Feel free to run others. Their results are interpreted in the same way as those shown here.

Let's continue with the `dataset` we have been using to measure the effect of nitrogen dioxide in the classroom on the reaction time of school children.

```
. use https://www.stata-press.com/data/r18/breathe, clear
(Nitrogen dioxide and attention)
```

We need to create the global macros that will hold our lists of continuous and factor-variable control variables:

```
. do https://www.stata-press.com/data/r18/no2
(output omitted)
```

To see how these lists were created, see [1.4 The primary dataset](#).

4.1 Interpreting standard incidence-rate ratios

If you are new to inferential lasso models and have not read [2.2 Fitting via cross-fit partialing out \(xpo\) using plugin](#), do that now. We will only explain how to interpret the incident-rate ratios below. Section 2.2 explains more.

Our count outcome is `omissions`, the number of times a student failed to respond to a stimulus while taking a test to measure reaction times. We are interested in how classroom nitrogen dioxide levels (`no2_class`) affect the number of omissions.

Our continuous controls are in the `global macro $cc`, and our `factor-variable` controls are in the `global macro $fc`, as they were in our very first example in section 2.2. We use `xpovpoisson` to fit the model,

```
. xpovpoisson omissions no2_class, controls($cc i.($fc)) rseed(12345)
(output omitted)
Cross-fit partialing-out      Number of obs      =      1,036
Poisson model                 Number of controls  =       32
                              Number of selected controls =      16
                              Number of folds in cross-fit =      10
                              Number of resamples      =       1
                              Wald chi2(1)              =      5.42
                              Prob > chi2              =      0.0199
```

omissions	IRR	Robust std. err.	z	P> z	[95% conf. interval]	
no2_class	1.022025	.0095654	2.33	0.020	1.003448	1.040946

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

We see that `xpovpoisson` reports an IRR (incidence-rate ratio) by default, rather than a coefficient. That is more useful for interpretation. The term “rate”, however, is less intuitive for the count of omissions. Often, counts are taken over a time and thus are considered rates. Our count is for a fixed-length test, so it is better to think of this as a ratio of means. Our point estimate of 1.02 means that we expect the number of omissions to go up by a factor of 1.02 for every unit increase in the level of nitrogen dioxide in the classroom. Our 95% confidence interval is 1.003 to 1.041, and the ratio is significantly different from 1 at the 5% level.

That rate might seem small, but the level of `no2_class` ranges from 7.8 to 52.6:

```
. summarize no2_class
```

Variable	Obs	Mean	Std. dev.	Min	Max
no2_class	1,089	30.16779	9.895886	7.794096	52.56397

The difference is over 44 micrograms per cubic meter. A reasonable question would be how much a student is affected in going from a classroom with, say, 8 micrograms to a classroom with 52 micrograms. `lincom` can answer that question if we tell it that we want IRRs reported:

```
. lincom _b[no2_class]*44, irr
( 1) 44*no2_class = 0
```

omissions	IRR	Std. err.	z	P> z	[95% conf. interval]	
(1)	2.608014	1.073998	2.33	0.020	1.163535	5.845752

The ratio is 2.6 and is significant, having exactly the same z-statistic as the original estimate. That is by construction because for the purpose of the test, we merely multiplied the underlying coefficient by a constant. A child is expected to make 2.6 times as many errors when exposed to 52 micrograms of nitrogen dioxide as compared with the number of errors when exposed to only 8 micrograms.

That result does not rely on the starting number of 8. It depends only on the difference. We could ask about the effect of adding 10 micrograms of nitrogen dioxide to whatever is the ambient level:

```
. lincom _b[no2_class]*10, irr
( 1) 10*no2_class = 0
```

omissions	IRR	Std. err.	z	P> z	[95% conf. interval]	
(1)	1.243414	.1163742	2.33	0.020	1.035023	1.493764

So adding 10 micrograms increases the expected number of omissions by 1.24. If the number of omissions was 4 before the increase, we expect just under 5 after. If it was 10, we expect 12.4 after.

Be careful if you want to take two steps of the 10-microgram increase. These are ratios, so a 20-microgram increase leads to a $1.24^2 = 1.54$ ratio.

We cannot estimate counts after any of the Poisson inferential lasso estimators. The theory for these estimators does not provide for estimating an intercept.

4.2 Interpreting models with factor variables

As we did with logit models for binary outcomes, let's run through a few of the examples that we first demonstrated with linear regression. We are going to set the models up quickly. Read sections 2.6 and 2.7 for more about the models. We will use the same tools; we will just ask them to provide IRRs.

Continuing with the same dataset, in [2.6 Fitting models with several variables of interest](#) we added age to our covariates of interest. That means we must pull age from our list of continuous controls:

```
. vl create cc41 = cc - (age)
note: $cc41 initialized with 9 variables.
```

As we did with logit models, we will use different global macro names throughout this section to avoid collisions with the original examples. Again, these globals hold the same variable lists—they just have a different name.

We fit the model.

```
. xpopoisson omissions no2_class age, controls($cc41 i.($fc)) rseed(12345)
(output omitted)
```

```
Cross-fit partialing-out      Number of obs          =       1,036
Poisson model                 Number of controls     =         31
                              Number of selected controls =         15
                              Number of folds in cross-fit =         10
                              Number of resamples      =          1
                              Wald chi2(2)             =       29.20
                              Prob > chi2             =       0.0000
```

omissions	IRR	Robust std. err.	z	P> z	[95% conf. interval]	
no2_class	1.023175	.005028	4.66	0.000	1.013368	1.033078
age	.8075872	.0406566	-4.24	0.000	.7317068	.8913366

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

We now have an IRR for `age` as well as for `no2_class`. They are both interpreted as we did `no2_class` above, which is to say, as you would any IRR.

In [2.7 Fitting models with factor variables of interest](#), we decided that we were interested in the effect of the child's grade in school and were no longer interested in nitrogen dioxide. Really, we just want to demonstrate a factor-variable covariate of interest.

We will set our controls to reflect this:

```
. vl create fc32 = fc - (grade)
note: $fc32 initialized with 7 variables.
. vl create cc32 = cc + (no2_class)
note: $cc32 initialized with 11 variables.
```

And we fit the `xpipoisson` model:

```
. xpipoisson omissions i.grade, controls($cc32 i.($fc32)) baselevels
> rseed(12345)
```

(output omitted)

```
Cross-fit partialing-out      Number of obs      =      1,036
Poisson model                 Number of controls  =         30
                              Number of selected controls =         11
                              Number of folds in cross-fit =         10
                              Number of resamples      =          1
                              Wald chi2(2)              =         4.74
                              Prob > chi2              =         0.0933
```

omissions	IRR	Robust std. err.	z	P> z	[95% conf. interval]	
grade						
2nd	1	(base)				
3rd	.6008938	.1451159	-2.11	0.035	.3743109	.9646349
4th	.443883	.1832475	-1.97	0.049	.197637	.9969392

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The expected number of omissions of third graders is 60% of that of second graders with a 95% CI of 0.37 to 0.96. Fourth graders have even fewer omissions. The point estimate is 44% of the number for second graders.

`contrast` works with IRRs just as it did with ORs in section 3.2. Again, we just need to add the option `eform(IRR)`.

```
. contrast ar.grade, eform(IRR)
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
grade			
(3rd vs 2nd)	1	4.45	0.0349
(4th vs 3rd)	1	1.21	0.2708
Joint	2	4.74	0.0933

	IRR	Std. err.	[95% conf. interval]	
grade				
(3rd vs 2nd)	.6008938	.1451159	.3743109	.9646349
(4th vs 3rd)	.7387046	.2031553	.4309005	1.266382

We specified reverse-adjacent (`ar`) contrasts, so comparisons will now be grade to grade rather than against a base grade. The first comparison is still between second and third grades and, of course, gives the same results as `xpoppoisson` itself.

The second comparison is between third and fourth grades. We fail to find a significant difference, though the point estimate is that fourth graders make only 74% of the omissions made by third graders.

As with the logit models, we will skip section 2.8 *Fitting models with interactions of interest* because it does not offer any new tools for analyzing odds ratios. You can run that model as an inferential lasso probit model on `omissions`. If you run any contrasts, be sure to add option `eform(IRR)`.

5 Exploring inferential model lassos

Aside from the two commands we have used in the examples in this entry, `[LASSO] lassoinfo` and `[LASSO] lassocoeff`, you are unlikely to need many of the postestimation commands commonly used after `lasso`. Regardless, most of them are available. You can create knot tables of coefficient selection, plot cross-validation functions, plot coefficient paths, display lasso coefficients, and even change the penalty parameter λ that is used to select controls.

See `[LASSO] lasso inference postestimation` for an overview and a list of postestimation commands that are available after the inferential lasso estimators. The entries for each command have examples that demonstrate their use after inferential lasso estimators.

6 Fitting an inferential model with endogenous covariates

We will replicate a well-known model that was used to illustrate a two-stage least squares estimator for handling an endogenous covariate; see Wooldridge (2010, ex. 5.3). Because the inferential lasso estimators provide variable selection that is robust to selection mistakes, we will introduce a flexible series expansion of the variables.

Wooldridge models the log of married women's wages (`lwage`) as a function of their experience (`exper`), the square of their experience, and their years of education (`educ`). Collectively, these are called exogenous covariates.

As is customary, education is treated as an endogenous variable. The reasoning is that we cannot measure innate ability, and ability is likely to influence both education level and income. Some disciplines refer to this as unobserved confounding rather than endogeneity. Either way, you cannot just run a regression of wages on education and experience and learn anything about the true effect of education on wages.

You need more information from variables that you presume are not affected by the woman's unmeasured ability—let's call them instruments. And, they also cannot belong in the model for wages. Wooldridge used their mothers' education (`motheduc`), their fathers' education (`fatheduc`), and their husbands' education (`huseduc`) as instruments for the woman's education. The instruments are also required to be exogenous, but we will just call them instruments.

The data are from [Mroz \(1987\)](#).

`xpovregress` and `poivregress` use lassos to select the exogenous covariates from a list of potential exogenous covariates. They use lassos to select the instruments from a set of potential instruments. This means we do not have to worry about introducing noise or weak instruments by possibly including irrelevant exogenous covariates or instruments. Lasso will ensure that sufficient amounts of irrelevant covariates are ignored. We are free to include the kitchen sink.

Let's add some variables that Wooldridge kept out. He was required to be thoughtful of introducing irrelevant covariates. We are not. To the list of potential exogenous covariates, we add the number of children younger than 6 (`kidslt6`), the number of children aged 6 or older (`kidsge6`), the women's ages (`age`), their husbands' ages (`husage`), and an indicator for living in an urban area (`citt`). We have nothing to add to the instruments. Good instruments are hard to find.

To make sure the sink is full, let's take all the exogenous variables and, instead of entering them only linearly, enter them as linear terms, as quadratic terms, and as all possible interactions. Let's do the same for our list of three instruments. This is often called a series expansion, or a Taylor-series expansion. It allows for nonlinearity in the way our exogenous covariates affect the outcome and in the way our instruments control endogeneity. We just did second-order expansion; you can go further.

We will continue using the variable-management tool `vl` to manage our lists of variables. First, we use the Mroz dataset and then create our base list of exogenous covariates and our base list of instruments.

```
. use https://www.stata-press.com/data/r18/mroz, clear
. vl create exogbase = (exper age husage kidslt6 kidsge6 city)
note: $exogbase initialized with 6 variables.
. vl create instbase = (motheduc fatheduc huseduc)
note: $instbase initialized with 3 variables.
```

The list of exogenous covariates is now in the [global macro](#) `$exogbase`, and the list of instruments is now in `$instbase`.

With these base lists in hand, we can perform our expansions to create flexible nonlinear forms:

```
. v1 substitute exog = c.exogbase c.exogbase#c.exogbase
. v1 substitute inst = c.instbase c.instbase#c.instbase
```

The # is the factor-variable operator for interaction. It can interact categorical variables, continuous variables, or both. We could have used it directly on our estimation command line, but those lines are already long enough. We also would have to handle macro expansion by typing `$exogbase` and such. `v1` already knows about `exogbase` and `instbase` and knows to handle them as lists. The `c.` prefix tells the # operator to treat the lists as continuous variables. # assumes categorical variables unless told otherwise.

Putting it all together, `c.exogbase` means to enter all the potential exogenous covariates as themselves (linearly). `c.exogbase#c.exogbase` means to enter all possible interactions of the variables. Because an interaction of a variable with itself is a quadratic, the quadratic (squared) terms get created as part of the expansion.

Let's look at the smaller of these two lists so that we can see what we have created:

```
. macro list inst
inst:      motheduc fatheduc huseduc c.motheduc#c.motheduc
          c.motheduc#c.fatheduc c.motheduc#c.huseduc
          c.fatheduc#c.fatheduc c.fatheduc#c.huseduc c.huseduc#c.huseduc
```

That is not too bad. We count nine terms—three linear terms and six interactions (including quadratic terms).

Macro `exog` has 27 terms.

Imagine what a third-order expansion would look like. You can run into the thousands of terms quickly.

Now we can use `xpovregress` to estimate the coefficient on the endogenous variable `educ`. We start with the plugin method to select the covariates. We do not have to specify plugin because it is the default. Specifying the rest of the model is easy because of the macro we created:

```
. xpovregress lwage (educ = $inst), controls($exog) rseed(12345)
Cross-fit fold 1 of 10 ...
Estimating lasso for lwage using plugin
Estimating lasso for educ using plugin
Cross-fit fold 2 of 10 ...
Estimating lasso for lwage using plugin
Estimating lasso for educ using plugin
(output omitted)
Cross-fit partialing-out      Number of obs      =      428
IV linear model              Number of controls =      27
                             Number of instruments =       9
                             Number of selected controls =       4
                             Number of selected instruments =       3
                             Number of folds in cross-fit =      10
                             Number of resamples =         1
                             Wald chi2(1) =         10.84
                             Prob > chi2 =         0.0010
```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
lwage						
educ	.0727853	.0221045	3.29	0.001	.0294612	.1161094

Endogenous: educ

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

In the header, we see that 4 of 27 controls were selected, and 3 of 9 possible instruments were selected. This is a sparse representation of the model.

We estimate that every year of education increases the log of wages by 0.073. Because wages are logged, we interpret that as a rate of change, so each year of education increases wages by 7.3%. That is close to Wooldridge's estimate of 8%, and his estimate is well within our 95% CI of 2.8% to 11.6%.

Let's see how the results compare if we select using cross-validation:

```
. xpoivregress lwage (educ = $inst), controls($exog) selection(cv) rseed(12345)
Cross-fit fold 1 of 10 ...
Estimating lasso for lwage using cv
Estimating lasso for educ using cv
Cross-fit fold 2 of 10 ...
Estimating lasso for lwage using cv
(output omitted)
Cross-fit partialing-out      Number of obs      =      428
IV linear model              Number of controls =      27
                             Number of instruments =      9
                             Number of selected controls =     20
                             Number of selected instruments =      7
                             Number of folds in cross-fit =     10
                             Number of resamples =      1
                             Wald chi2(1) =      7.68
                             Prob > chi2 =      0.0056
```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
lwage						
educ	.0645424	.0232832	2.77	0.006	.0189082	.1101765

Endogenous: educ

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

Cross-validation selected 20 controls compared with the 4 selected by plugin. It selected 7 instruments compared with the 3 selected by plugin. Our point estimate of the change in wages for each additional year of education is 6.5% with a CI of 1.9% to 11.0%. The coefficient estimate from both cross-validation and plugin are significant at the 5% level. Despite having slightly different coefficient estimates, plugin and cross-validation lead to the same inferences.

References

- Mroz, T. A. 1987. The sensitivity of an empirical model of married women's hours of work to economic and statistical assumptions. *Econometrica* 55: 765–799. <https://doi.org/10.2307/1911029>.
- Sunyer, J., E. Suades-González, R. García-Esteban, I. Rivas, J. Pujol, M. Alvarez-Pedrerol, J. Forn, X. Querol, and X. Basagaña. 2017. Traffic-related air pollution and attention in primary school children: Short-term association. *Epidemiology* 28: 181–189. <https://doi.org/10.1097/EDE.0000000000000603>.
- Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

Also see

[LASSO] [Lasso intro](#) — Introduction to lasso

[LASSO] [Lasso inference intro](#) — Introduction to inferential lasso models

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](#).