Title

stata.com

telasso — Treatment-effects estimation using lasso

Description

`telasso` estimates the average treatment effect (ATE), the average treatment effect on the treated (ATET), and the potential-outcome means (POMs) from observational data by augmented inverse-probability weighting (AIPW) while using lasso methods to select from potential control variables to be included in the model.

AIPW estimators combine aspects of regression-adjustment and inverse-probability-weighted methods. AIPW estimators have the double-robust property.

`telasso` accepts a continuous, binary, count, or nonnegative outcome.

See [CAUSAL] **teffects intro** or [CAUSAL] **teffects intro advanced** for more information about estimating treatment effects from observational data. See [LASSO] **Lasso inference intro** for more information on estimating coefficients and standard errors for a subset of variables while using lasso methods to select from a set of control variables.

Quick start

ATE of binary treatment `treat` using a linear model for outcome `y1` on `x1`–`x100` and a logistic model for `treat` on `w1`–`w100`; use lassos to select variables from `x1`–`x100` for the outcome model and from `w1`–`w100` for the treatment model

```
telasso (y1 x1-x100) (treat w1-w100)
```

Same as above, but estimate the ATET

```
telasso (y1 x1-x100) (treat w1-w100), atet
```

Use a Poisson model for count outcome `y2`

```
telasso (y2 x1-x100, poisson) (treat w1-w100)
```

Same as above, but use a probit model for `treat`

```
telasso (y2 x1-x100, poisson) (treat w1-w100, probit)
```

Use BIC instead of a plugin iterative formula to select the optimal λ^* in each lasso

```
telasso (y1 x1-x100) (treat w1-w100), selection(bic)
```

Perform cross-fitting with five folds

```
telasso (y1 x1-x100) (treat w1-w100), xfolds(5)
```

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

```
telasso (y1 x1-x100) (treat w1-w100), xfolds(5) resample(15)
```

Use BIC to select covariates in the lasso for `y1` for treatment level 1 only

```
telasso (y1 x1-x100, lasso(1, selection(bic)) (treat w1-w100)
```

Use cross-validation (CV) for the lasso for `treat` only

```
telasso (y1 x1-x100) (treat w1-w100, lasso(selection(cv))
```

Menu

Statistics > Causal inference/treatment effects > Continuous outcomes > AIPW with lasso covariate selection

Statistics > Causal inference/treatment effects > Binary outcomes > AIPW with lasso covariate selection

Statistics > Causal inference/treatment effects > Count outcomes > AIPW with lasso covariate selection

Statistics > Causal inference/treatment effects > Nonnegative outcomes > AIPW with lasso covariate selection

Syntax

```
telasso (ovar omvarlist [, omodel om_options])
  (tvar tmvarlist [, tmodel tm_options]) [if] [in] [weight] [, stat options]
```

ovar is a binary, count, continuous, or nonnegative outcome of interest.

omvarlist specifies the covariates in the outcome model.

tvar must contain a binary value representing the treatment levels.

tmvarlist specifies the covariates in the treatment model.

omodel	Description
Model	
`linear`	linear outcome model; the default
`logit`	logistic outcome model
`probit`	probit outcome model
`poisson`	exponential-mean outcome model

om_options	Description
Model	
`lasso(`[`#,`] *lasso_options*`)`	specify options for the lassos for *ovar* at treatment level #; may be repeated
* `sqrtlasso`	use square-root lassos instead of lassos for *ovar*
* `sqrtlasso(`[`#,`] *lasso_options*`)`	specify options for the square-root lassos for *ovar* at treatment level #; may be repeated
`noconstant`	suppress constant term
`ainclude(`*varlist*`)`	specify variables that should always be included in the outcome model
`noselection`	suppress model selection for the outcome model

*`sqrtlasso()` may be specified only when *omodel* is `linear`.

tmodel	Description
Model	
`logit`	logistic treatment model; the default
`probit`	probit treatment model

tm_options	Description
Model	
`lasso(`*lasso_options*`)`	specify options for the lasso for *tvar*
`noconstant`	suppress constant term
`ainclude(`*varlist*`)`	specify variables that should always be included in the treatment model
`noselection`	suppress model selection for the treatment model

stat	Description
Stat	
`ate`	estimate average treatment effect in population; the default
`atet`	estimate average treatment effect on the treated
`pomeans`	estimate potential-outcome means

options	Description
Selection	
`selection(plugin)`	use a plugin iterative formula to select an optimal value of the lasso penalty parameter λ^* for each lasso; the default
`selection(cv)`	use CV to select an optimal value of the lasso penalty parameter λ^* for each lasso
`selection(adaptive)`	use adaptive lasso to select an optimal value of the lasso penalty parameter λ^* for each lasso
`selection(bic)`	use BIC to select an optimal value of the lasso penalty parameter λ^* for each lasso
`xfolds[(#)]`	use # folds for cross-fitting
`resample[(#)]`	repeat sample splitting # times and average results
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `robust` or `cluster` *clustvar*
Reporting	
`level(#)`	set confidence level; default is `level(95)`
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Optimization	
`rseed(#)`	set random-number seed
`verbose`	display a verbose iteration log
`[no]log`	display or suppress an iteration log

Advanced

pstolerance(#)	set tolerance for overlap assumption
osample(*newvar*)	*newvar* identifies observations that violate the overlap assumption
control(# | *label*)	specify the level of *tvar* that is the control
reestimate	refit the model after using lassoselect to select a different λ^*
noheader	do not display the header on the coefficient table
coeflegend	display legend instead of statistics

omvarlist and *tmvarlist* may contain factor variables; see [U] **11.4.3 Factor variables**.

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

Default weights are not allowed. iweights are allowed when selection(plugin), selection(adaptive), selection(cv), or selection(bic) is specified. fweights are allowed when selection(plugin) or selection(bic) is specified. See [U] **11.1.6 weight**.

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

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

Options

Model

Options specific to the outcome model (*om_options*) are the following:

lasso(⌊#,⌋ *lasso_options*) lets you set different options for different lassos or set advanced options for all lassos for the outcome model. Specify a # followed by the options to apply settings to the lasso of *ovar* at treatment level # only. This option is repeatable as long as a different treatment level is given in each specification. If # is not specified, the lasso options apply to lassos of *ovar* at all treatment levels. *lasso_options* are selection(), grid(), stop(), cvtolerance(), bictolerance(), tolerance(), and dtolerance(); see [LASSO] **lasso options**. When lasso(⌊#,⌋ selection()) is specified, it overrides any global selection() option for the outcome variable at the specified treatment level.

sqrtlasso specifies that square-root lassos rather than lassos should be used in selecting variables in models for the outcome at both treatment levels. This option is not allowed if *omodel* is logit, probit, or poisson.

sqrtlasso(⌊#,⌋ *lasso_options*) works like the option lasso(), except square-root lassos for the outcome variable are done rather than regular lassos. This option is not allowed if *omodel* is logit, probit, or poisson. Specify a # followed by the options to apply settings to the lasso of *ovar* at treatment level # only. This option is repeatable as long as a different treatment level is given in each specification. If # is not specified, the lasso options apply to lassos of *ovar* at all treatment levels. *lasso_options* are selection(), grid(), stop(), cvtolerance(), bictolerance(), tolerance(), and dtolerance(); see [LASSO] **lasso options**. When sqrtlasso(⌊#,⌋ selection()) is specified, it overrides any global selection() option for the outcome variable at the specified treatment level.

noconstant suppresses the constant term in the outcome model.

ainclude(*varlist*) specifies variables that are always included in the outcome model. Lassos will choose to include or exclude other variables in *omvarlist*.

noselection specifies not to perform model selection in the outcome model.

Options specific to the treatment model (*tm_options*) are the following:

`lasso(`*lasso_options*`)` lets you set advanced options for the lasso for the treatment model. *lasso_options* are `selection()`, `grid()`, `stop()`, `cvtolerance()`, `bictolerance()`, `tolerance()`, and `dtolerance()`; see [LASSO] **lasso options**. When `lasso(selection())` is specified, it overrides any global `selection()` option for the treatment variable.

`noconstant` suppresses the constant term in the treatment model.

`ainclude(`*varlist*`)` specifies variables that are always included in the treatment model. Lassos will choose to include or exclude other variables in *tmvarlist*.

`noselection` specifies not to perform model selection in the treatment model.

Selection

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

`selection(plugin)` is the default. It selects λ^* based on a "plugin" iterative formula dependent on the data. See [LASSO] **lasso options**.

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

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

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

`xfolds[(#)]` specifies the number of folds for cross-fitting. Not specifying `xfolds` or `xfolds(#)` is equivalent to specifying `xfolds(1)`. In other words, by default no cross-fitting is done. Specifying `xfolds` alone is equivalent to specifying `xfolds(10)`; that is, cross-fitting is done by randomly dividing the original data into 10 folds.

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

SE/Robust

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

Reporting

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

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

Optimization

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

verbose displays a verbose log showing the iterations of each lasso estimation. This option is useful when doing selection(cv) or selection(adaptive). It allows monitoring the progress of the lasso estimations for these selection methods, which can be time consuming when there are many variables specified in *omvarlist* or *tmvarlist*.

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

Advanced

pstolerance(#) specifies the tolerance used to check the overlap assumption. The default value is pstolerance(1e-5). telasso will exit with an error if an observation has an estimated propensity score smaller than that specified by pstolerance().

osample(*newvar*) specifies that indicator variable *newvar* be created to identify observations that violate the overlap assumption.

control(# | *label*) specifies the level of *tvar* that is the control. The default is the first treatment level. You may specify the numeric level # (a nonnegative integer) or the label associated with the numeric level. control() may not be specified with statistic pomeans.

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

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

reestimate may be combined only with reporting options.

noheader prevents the coefficient table header from being displayed.

coeflegend; see [R] Estimation options.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Overview

telasso estimates treatment effects while using lasso techniques to select the variables in the model. Like the lasso for inference commands, inferences drawn about the ATEs, ATETs, and POMs reported by telasso are robust to model selection mistakes made by lasso. In addition, like the teffects aipw command, the estimated effects are consistent if the functional form of only one of the outcome or treatment model is correctly specified. Thus, telasso is a version of the AIPW estimator that also allows for covariate selection via lasso and is robust to functional form misspecification in one of the models as well as errors in selection of covariates by lasso. It is, however, required that the true covariates are a subset of the specified list of potential covariates. See [CAUSAL] **teffects intro** or [CAUSAL] **teffects intro advanced** for an introduction of treatment effects estimation. See [CAUSAL] **teffects aipw** for more information on the AIPW estimator. See [LASSO] **Lasso intro** for an introduction to lasso. See [LASSO] **Lasso inference intro** for an introduction to making inference after using lasso for model selection.

Researchers who have datasets with many variables, sometimes more than observations, face difficult decisions. They cannot use all the variables in their dataset as covariates in a model, and therefore, they may want to use a model selection method such as lasso to select covariates. In addition, rather than simply allowing covariates to enter a model linearly, researchers may want to fit a more flexible and realistic model to their data by including higher-order terms, interactions, spline basis functions, and the like, but this can also produce more terms than can be included in a model and require a method such as lasso for selection. The lasso methods used in telasso allow for making valid inference after selecting from among the potential covariates.

In the context of treatment-effect estimation, why do we need model selection? It is because of the intrinsic conflicts between two crucial assumptions used to identify treatment effects: conditional independence (CI) and overlap assumptions. CI means that, dependent on a set of control variables, the potential outcome is independent of the treatment assignment. The more variables there are in the model, the more plausible it is that the CI assumption is satisfied. On the other hand, the overlap assumption implies that there is always a positive probability that any given unit is treated or untreated. The fewer variables there are in the model, the more comfortable we can be that the overlap assumption is satisfied. To summarize, the conflict is that the CI assumption expects many variables in the model, but the overlap assumption expects few variables. By including many potential variables in the model and allowing lasso to select from among them, we can reconcile this conflict. For a more detailed discussion, see Farrell (2015) and Chernozhukov et al. (2018).

Model selection, however, does not come for free. If researchers use model selection but conduct inference ignoring the fact that they did model selection, the inference results are possibly wrong. This is because model selection techniques make mistakes. Making inferences without considering the variability in model selection is a dangerous practice (see Leeb and Pötscher [2005] and Leeb and Pötscher [2006]). Instead, the inference should be robust to model selection mistakes.

Recently, Farrell (2015) and Chernozhukov et al. (2018) proposed that modern model selection techniques, such as lasso methods for inference, be combined with the doubly robust AIPW estimator in Rosenbaum and Rubin (1983) to estimate the ATEs. The intuition behind this method is twofold. On one hand, lasso techniques resolve the conflicts between the CI and overlap assumptions. Although CI assumption expects many variables, we only need the covariates that matter for the outcome. If lasso selects a sparse model, the overlap assumption is more plausible to be satisfied. On the other hand, the proposed estimator should guard against model selection mistakes. The doubly robust AIPW estimator happens to satisfy this property. Double robustness also means that the estimates of treatment effects are consistent if either the outcome model or the treatment model is correctly specified.

telasso implements estimators in Chernozhukov et al. (2018) for the ATEs, ATETs, and POMs from observational data.

The telasso estimator uses a five-step approach to estimating ATE:

1. It uses lasso techniques to select variables in the outcome model for each treatment level.
2. Based on the selected variables in step 1, it fits separate regression models of the outcome for each treatment level and obtains the treatment-specific predicted outcomes for each subject.
3. It uses lasso techniques to select variables in the treatment model.
4. Based on the selected variables in step 3, it estimates the parameters of the treatment model and computes inverse-probability weights.
5. It computes the weighted means of the treatment-specific predicted outcomes, where the weights are the inverse-probability weights computed in step 4. The contrasts of these weighted averages provide the estimates of the ATE.

Steps 1 and 3 perform the model selection for the outcome and treatment models, respectively. Using the selected variables, steps 2, 4, and 5 construct moment conditions to estimating ATEs. The resulting estimator is consistent under CI, overlap, and independent and identically distributed assumptions. The inference is robust to the mild model selection mistakes that could happen in steps 1 and 3. This estimator is also robust to model misspecification in either the outcome or the treatment model because of the double robust moment condition used in step 5.

telasso also implements the double machine learning estimator in Chernozhukov et al. (2018) for ATEs, ATETs, and POMs. Double machine learning relaxes the sparsity assumption needed for lasso methods. The sparsity assumption implies that lasso can only have good properties if it selects a few variables from a potentially large number of candidates. Double machine learning allows telasso to select more variables in the models and still be valid.

telasso allows the outcome variable to be modeled using a linear, logistic, probit, or Poisson model. For a linear outcome model, square-root lasso can be used instead of lasso in step 1.

Estimating the ATE with lassos for covariate selection

In some cases, we want to estimate a treatment effect when we have several variables in our dataset that, when interacted, create a large set of candidate covariates in our model.

▷ Example 1: ATE of bilateral lung transplant

We first illustrate telasso with an example that compares two types of lung transplants. Bilateral lung transplant (BLT) is usually associated with a higher death rate in the short term after the operation but with a more significant improvement in the quality of life compared with the single lung transplant (SLT). As a result, for patients who need to decide between these two treatment options, knowing the effect of BLT (versus SLT) on quality of life is essential. We can measure quality of life based on an individual's forced expiratory volume in one second (FEV1).

We have a fictional dataset (lung.dta) inspired by Koch, Vock, and Wolfson (2018). The outcome (fev1p) is FEV1% measured one year after the operation. FEV1% is the percentage of FEV1 that the patient has relative to a healthy person with similar characteristics. The treatment variable (transtype) indicates whether the treatment is BLT or SLT.

To open the dataset and describe it, we type

```
. use https://www.stata-press.com/data/r18/lung
(Fictional data on lung transplant)
. describe *, short
```

Variable name	Storage type	Display format	Value label	Variable label
agep	byte	%10.0g		Patient age (years)
bmip	double	%10.0g		Patient body mass index
diabetesp	byte	%12.0g	lbdiab	Patient diabetes status
heightp	double	%10.0g		Patient height (cm)
o2amt	double	%10.0g		Oxygen delivered
karn	byte	%8.0g	lbyes	Karnofsky score > 60
lungals	double	%10.0g		Lung allocation score
racep	byte	%8.0g	lbrace	Patient race
sexp	byte	%8.0g	lbsex	Patient gender
lifesvent	byte	%8.0g	lbyes	Life support ventilator needed
assisvent	byte	%8.0g	lbyes	Assisted ventilation needed
centervol	double	%10.0g		Center volume
walkdist	double	%10.0g		Walking distance in 6 minutes
o2rest	byte	%8.0g	lbyes	Oxygen needed at rest
aged	byte	%10.0g		Donor age (years)
raced	byte	%8.0g	lbrace	Donor race
bmid	double	%10.0g		Donor body mass index
smoked	byte	%8.0g	lbyes	Donor if has history of smoking
cmv	byte	%8.0g	lbyes	Positive cytomegalovirus test
deathcause	byte	%8.0g	lbyes	Cause of death - traumatic brain injury
diabetesd	byte	%12.0g	lbdiab	Donor diabetes status
expandd	byte	%8.0g	lbyes	Expanded donor needed
heightd	double	%10.0g		Donor height (cm)
sexd	byte	%8.0g	lbsex	Donor gender
distd	int	%10.0g		Donor to treatment center distance
lungpo2	double	%10.0g		Lung PO2
lungalloc	byte	%8.0g	lballo	Lung allocation status
hratio	double	%10.0g		Height ratio
ischemict	double	%10.0g		Ischemic time
genderm	byte	%19.0g	lbgm	Matching gender status
racem	byte	%17.0g	lbrm	Matching race status
transtype	byte	%8.0g	lbtau	Lung transplant type
fev1p	double	%10.0g		Percentage of predicted value of FEV1

Thirty-one variables measure some characteristics of the patients and donors. To construct control variables, we want to use these variables and the interactions among them. It would be tedious to type these variable names one by one to distinguish between continuous and categorical variables. `vl` is a suite of commands that simplifies this process.

First, we use `vl set` to automatically partition the variables into continuous and categorical variables. The global macro `$vlcategorical` contains all the categorical variable names, and `$vlcontinuous` contains all the continuous variable names.

```
. vl set
```

Macro	Macro's contents	
	# Vars	Description
System		
$vlcategorical	18	categorical variables
$vlcontinuous	13	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. *vlnames* are global macros. Type the *vlname* 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*.

```
. display "$vlcontinuous"
bmip heightp o2amt lungals centervol walkdist bmid heightd distd lungpo2
> hratio ischemict fev1p
. display "$vlcategorical"
diabetesp karn racep sexp lifesvent assisvent o2rest raced smoked cmv
> deathcause diabetesd expandd sexd lungalloc genderm racem transtype
```

Second, we use `vl create` to create customized variable lists. Specifically, `$cvars` contains all the continuous variables except the outcome (`fev1p`), and `$fvars` consists of all the categorical variables except the treatment (`transtype`). Finally, `vl sub` substitutes the global macro `$allvars` with the full second-order interaction between the continuous variables in `$cvars` and categorical variables in `$fvars`. We will use `$allvars` as the control variables for both outcome model and treatment model.

```
. vl create cvars = vlcontinuous - (fev1p)
note: $cvars initialized with 12 variables.
. vl create fvars = vlcategorical - (transtype)
note: $fvars initialized with 17 variables.
. vl sub allvars = c.cvars i.fvars c.cvars#i.fvars
```

Now we are ready to use `telasso` to estimate the ATEs. We assume a linear outcome model and a logit treatment model, the defaults for `telasso`. We type

```
. telasso (fev1p $allvars)  (transtype $allvars)
Estimating lasso for outcome fev1p if transtype = 0 using plugin method ...
Estimating lasso for outcome fev1p if transtype = 1 using plugin method ...
Estimating lasso for treatment transtype using plugin method ...
Estimating ATE ...
Treatment-effects lasso estimation    Number of observations      =        937
Outcome model:   linear               Number of controls          =        454
Treatment model: logit                Number of selected controls =          8
```

fev1p	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
transtype (BLT vs SLT)	37.51841	.1606703	233.51	0.000	37.20351	37.83332
POmean						
transtype SLT	46.4938	.2021582	229.99	0.000	46.09757	46.89002

The FEV1% if all the patients were to choose BLT is expected to be 38 percentage points higher than the 46% average expected if all patients were to choose a SLT. Among the 454 control variables, `telasso` selects only 8 of them. To summarize the model selection for both the outcome and the treatment models, we can use `lassoinfo`.

```
. lassoinfo
    Estimate: active
     Command: telasso
```

Variable	Model	Selection method	lambda	No. of selected variables
fev1p				
transt~e ~0	linear	plugin	.2239121	5
transt~e ~1	linear	plugin	.1986153	6
transtype	logit	plugin	.0748279	3

If we want to see which variables are selected by each lasso, we can use `lassocoef`. Notice that there are two lassos associated with the outcome of `fev1p`. One is for outcome `fev1p` when the treatment `transtype` is 0, and the other is for `fev1p` when `transtype` is 1. So, in `lassocoef`, we need to use the options `tlevel()` and `for()` to refer to the lasso for the outcome variable at a specified treatment level. In contrast, for the treatment of `transtype`, there is only one lasso; therefore, we only need to use option `for()` to specify the lasso for the treatment variable.

```
. lassocoef (., for(fev1p) tlevel(0)) (., for(fev1p) tlevel(1))
> (., for(transtype))
```

	fev1p(0)	fev1p(1)	transtype
heightp	x	x	
centervol	x	x	
walkdist	x	x	x
lungpo2	x	x	x
diabetesd#c.lungpo2			
0	x		
diabetesp#c.walkdist			
0		x	
assisvent#c.walkdist			
0		x	
ischemict			x
_cons	x	x	x

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

The lassos selected `walkdist` and `lungpo2` in all three models, while `heightp`, `centervol`, `0.diabetesd#c.lungpo2`, `0.diabetesp#c.walkdist`, `0.assisvent#c.walkdist`, and `ischemict` were each selected as covariates in one or two models.

◁

Choosing the tuning parameter

By default, telasso uses a plugin method to choose the tuning parameter λ in the lasso steps. We can also use BIC, cross-validation, or adaptive lasso to select the optimal λ.

▷ Example 2: Choosing λ via BIC

Here we use the selection(bic) option to select λ by minimizing BIC.

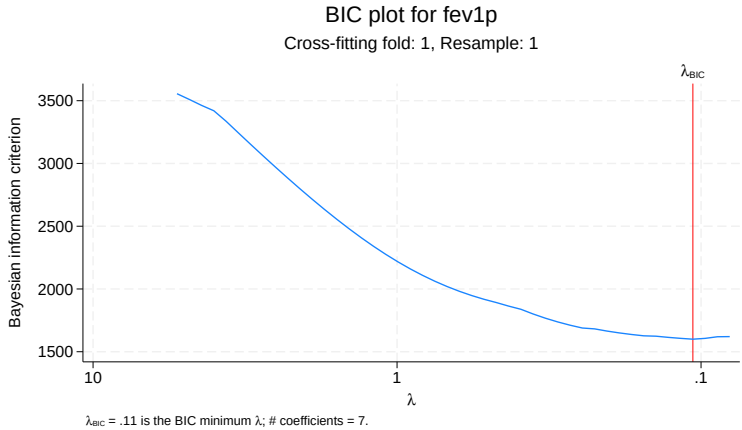
```
. telasso (fev1p $allvars)  (transtype $allvars), selection(bic)

Estimating lasso for outcome fev1p if transtype = 0 using BIC ...
Estimating lasso for outcome fev1p if transtype = 1 using BIC ...
Estimating lasso for treatment transtype using BIC ...
Estimating ATE ...

Treatment-effects lasso estimation    Number of observations      =        937
Outcome model:   linear               Number of controls          =        454
Treatment model: logit                Number of selected controls =         18
```

fev1p	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
transtype (BLT vs SLT)	37.54872	.2222001	168.99	0.000	37.11322	37.98423
POmean						
transtype SLT	46.44739	.2282797	203.47	0.000	45.99997	46.89481

We can interpret the estimation results in a similar way as in example 1. The FEV1% if all the patients were to choose BLT is expected to be 38 percentage points higher than the average of 46% that would be expected if all patients were to choose a SLT. This result is similar to example 1, where we used the default plugin method to select the tuning parameter. However, among 454 controls, telasso with BIC selects 18 of them, which is more than the plugin method selected.

We can use `bicplot` for a global view about how the BIC function changes as the λ's change. Here we show the BIC plot for the outcome `fev1p` when treatment `transtype` is 1. In `bicplot`, we use options `for()` and `tlevel()` to refer to the lasso for the outcome variable at a specified treatment level.

```
. bicplot, for(fev1p) tlevel(1)
```



From this plot, we can see that the optimal $\lambda = 0.11$ is chosen by the minimum BIC. If we want to investigate this further, we can use `lassoknots` to see which variables are selected or dropped for each λ.

```
. lassoknots, for(fev1p) tlevel(1) display(nonzero var bic)
```

ID	lambda	No. of nonzero coef.	BIC	Variables (A)dded, (R)emoved, or left (U)nchanged
2	4.823808	1	3510.118	A walkdist
5	3.649034	2	3335.87	A lungpo2
28	.4294227	3	1863.595	A heightp
29	.391274	4	1838.266	A centervol
35	.2239014	6	1682.065	A 0.diabetesp#c.walkdist 0.assisvent#c.walkdist
40	.1406166	7	1623.74	A heightd
* 43	.1063713	7	1599.977	U
44	.0969216	9	1606.419	A 0.karn#c.walkdist 0.raced#c.lungpo2
45	.0883113	12	1619.473	A 0.sexd#c.centervol 1.racep#c.centervol 0.deathcause#c.centervol
46	.080466	13	1620.666	A 1.lungalloc#c.lungpo2

```
* lambda selected by Bayesian information criterion.
```

We see that the 43rd λ, with value 0.1064, minimizes the BIC function, and there are seven selected variables at this λ.

◁

Estimating the ATET

Sometimes, we want to estimate the ATETs to determine the effect on those who actually received the treatment.

▷ Example 3: ATET for BLT

We use the `atet` option to estimate the ATETs. We type

```
. telasso (fev1p $allvars)  (transtype $allvars), atet
Estimating lasso for outcome fev1p if transtype = 0 using plugin method ...
Estimating lasso for outcome fev1p if transtype = 1 using plugin method ...
Estimating lasso for treatment transtype using plugin method ...
Estimating ATET ...
Treatment-effects lasso estimation    Number of observations      =        937
Outcome model:   linear               Number of controls          =        454
Treatment model: logit                Number of selected controls =          8
```

fev1p	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATET						
transtype (BLT vs SLT)	35.78157	.1831478	195.37	0.000	35.42261	36.14053
POmean						
transtype SLT	43.35214	1.268976	34.16	0.000	40.86499	45.83929

For the patients who have a BLT, we expect the average FEV1% to be 36 percentage points higher than if all of them choose an SLT.

◁

High-dimensional semiparametric models

Sometimes, the theory suggests that some variables are essential to control the confounders, but it is silent on the functional form of the model. In the next example, we will illustrate how to use `telasso` to estimate the ATE in a high-dimensional semiparametric model.

▷ Example 4: ATE of 401(k) eligibility

We want to estimate the effect of 401(k) eligibility (`e401`) on net financial assets (`asset`) using data reported by Chernozhukov and Hansen (2004). These data are from a sample of households in the 1990 Survey of Income and Program Participation (SIPP). The data contain information on the head of the household: income (`income`), age (`age`), years of education (`educ`), whether to receive pension benefit (`pension`), marital status (`married`), and whether to participate in the IRA (`ira`).

One concern when determining the effect of 401(k) eligibility on financial assets is that choosing to work for a company that offers a 401(k) plan is not randomly assigned. To overcome this issue, Poterba, Venti, and Wise (1994, 1995) propose that after conditioning on income, we can take working for a company that offers a 401(k) as exogenous.

We want to include income and age as covariates in the model, but we do not want to assume that they enter the model linearly. Because we want a more flexible model, we use makespline to create B-splines of order 3 with 3 knots at percentiles of variables income and age. The generated variables will be used to form a semiparametric model. These variables have the common stub _bs for easy reference later.

```
. use https://www.stata-press.com/data/r18/assets, clear
(Excerpt from Chernozhukov and Hansen (2004))
. makespline bspline income age, basis(_bs) knots(3)
```

We describe the generated B-spline variables. Seven terms are generated for each of the original variables.

```
. describe _bs*
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
_bs_1_1         double  %10.0g                B-spline basis term 1 for income
_bs_1_2         double  %10.0g                B-spline basis term 2 for income
_bs_1_3         double  %10.0g                B-spline basis term 3 for income
_bs_1_4         double  %10.0g                B-spline basis term 4 for income
_bs_1_5         double  %10.0g                B-spline basis term 5 for income
_bs_1_6         double  %10.0g                B-spline basis term 6 for income
_bs_1_7         double  %10.0g                B-spline basis term 7 for income
_bs_2_1         double  %10.0g                B-spline basis term 1 for age
_bs_2_2         double  %10.0g                B-spline basis term 2 for age
_bs_2_3         double  %10.0g                B-spline basis term 3 for age
_bs_2_4         double  %10.0g                B-spline basis term 4 for age
_bs_2_5         double  %10.0g                B-spline basis term 5 for age
_bs_2_6         double  %10.0g                B-spline basis term 6 for age
_bs_2_7         double  %10.0g                B-spline basis term 7 for age
```

Next, we define the control variables as the categorical variables (pension, married, and ira), the generated spline variables (_bs*), and all interactions among these variables. The global macro $controls contains the defined control variables, and we will use it in both the outcome and the treatment model.

```
. global vars c.(_bs*) i.(pension married ira)
. global controls $vars ($vars)#($vars)
```

Now, we are ready to use telasso to estimate the ATEs of 401(k) eligibility on net financial assets. We use a linear nonparametric series to approximate the outcome model. Moreover, we use a logit nonparametric series to approximate the treatment model. The global macro $controls defines terms in the nonparametric series.

```
. telasso (assets $controls) (e401 $controls), rseed(111)
Estimating lasso for outcome assets if e401k = 0 using plugin method ...
Estimating lasso for outcome assets if e401k = 1 using plugin method ...
Estimating lasso for treatment e401k using plugin method ...
Estimating ATE ...
Treatment-effects lasso estimation    Number of observations      =      9,913
Outcome model:   linear               Number of controls          =        221
Treatment model: logit                Number of selected controls =         48
```

assets	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
e401k (Eligible vs Not elig..)	8391.237	1136.779	7.38	0.000	6163.191	10619.28
POmean						
e401k Not eligi..	13853.67	796.7734	17.39	0.000	12292.03	15415.32

The net financial assets if all the workers work for the companies with a 401(k) plan is expected to be $8,391 more than the average of $13,854 that is expected if all the workers work for the companies without a 401(k) plan.

◁

Stored results

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

Scalars

`e(N)`	number of observations
`e(N0)`	number of observations for treatment level 0
`e(N1)`	number of observations for treatment level 1
`e(N_clust)`	number of clusters
`e(k_omvars)`	number of potential control variables in the outcome model
`e(k_omvars_sel)`	number of selected control variables in the outcome model
`e(k_tmvars)`	number of potential control variables in the treatment model
`e(k_tmvars_sel)`	number of selected control variables in the treatment model
`e(k_controls)`	number of potential control variables in the outcome and treatment models
`e(k_controls_sel)`	number of selected control variables in the outcome and treatment models
`e(k_levels)`	number of levels in treatment variable
`e(n_xfolds)`	number of folds for cross-fitting
`e(n_resample)`	number of resamples
`e(treated)`	level of treatment variable defined as treated
`e(control)`	level of treatment variable defined as control
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`telasso`
`e(cmdline)`	command as typed
`e(depvar)`	name of outcome variable
`e(tvar)`	name of treatment variable
`e(tmodel)`	`logit` or `probit`
`e(omodel)`	`linear`, `logit`, `probit`, or `poisson`
`e(omvars)`	potential control variables in the outcome model
`e(omvars_sel)`	selected control variables in the outcome model

e(tmvars)	potential control variables in the treatment model
e(tmvars_sel)	selected control variables in the treatment model
e(stat)	statistic estimated: ate, atet, or pomeans
e(wtype)	weight type
e(wexp)	weight expression
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(tlevels)	levels of treatment variable
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(rngstate)	random-number state used
e(properties)	b V
e(predict)	program used to implement predict
e(select_cmd)	program used to implement lassoselect
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

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

Functions

e(sample)	marks estimation sample

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

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

The model

We consider estimating the ATEs, ATETs, and POMs when the treatment effects are heterogeneous and the treatment is binary. For notational simplicity, we drop the subscript i indicating the ith observation.

The outcome model is

$$y = g_0(\tau, \mathbf{x}) + u \qquad E(u|\tau, \mathbf{x}) = 0$$

where y is the outcome variable, τ is the binary treatment variable, $\mathbf{x}$ are potentially high-dimensional control variables in the outcome model, and $g_0(\tau, \mathbf{x})$ is the expected potential outcome given a level of treatment and covariates $\mathbf{x}$. Because each unit can only be treated or not treated, the observed outcome y can only be one of $g_0(1, \mathbf{x}) + u$ or $g_0(0, \mathbf{x}) + u$.

The treatment model is

$$\tau = m_0(\mathbf{z}) + v \qquad E(v|\mathbf{z}) = 0$$

where $\mathbf{z}$ are potentially high-dimensional covariates in the treatment model. $m_0(\mathbf{z})$ is the expected value of τ given $\mathbf{z}$. In other words, $m_0(\mathbf{z})$ is the probability of a unit getting treated given $\mathbf{z}$.

The parameter of interest θ_0 is ATE, ATET, or POMs.

ATE is

$$\theta_0 = E\left\{g_0(1,\mathbf{x}) - g_0(0,\mathbf{x})\right\}$$

ATET is

$$\theta_0 = E\left\{g_0(1,\mathbf{x}) - g_0(0,\mathbf{x})|\tau = 1\right\}$$

POMs when $\tau = 1$ is

$$\theta_0 = E\left\{g_0(1,\mathbf{x})\right\}$$

POMs when $\tau = 0$ is

$$\theta_0 = E\left\{g_0(0,\mathbf{x})\right\}$$

We have several remarks about the above model.

1. The outcome model can be one of `linear`, `logit`, `probit`, or `poisson`. Let $\boldsymbol{\beta}_t$ be the outcome model parameters when treatment $\tau = t$. The table below provides details about the available functional form of $g_0(\tau, \mathbf{x})$.

Outcome model	Functional form for $g_0(\tau = t, \mathbf{x})$
`linear`	$\mathbf{x}'\boldsymbol{\beta}_t$
`logit`	$\exp(\mathbf{x}'\boldsymbol{\beta}_t)/\{1 + \exp(\mathbf{x}'\boldsymbol{\beta}_t)\}$
`probit`	$\Phi(\mathbf{x}'\boldsymbol{\beta}_t)$
`poisson`	$\exp(\mathbf{x}'\boldsymbol{\beta}_t)$

2. The treatment model can be either `logit` or `probit`. Let γ be the parameters in the treatment model. The table below provides details about the available functional form of $m_0(\mathbf{z})$.

Treatment model	Functional form for $m_0(\mathbf{z})$
`logit`	$\exp(\mathbf{z}'\boldsymbol{\gamma})/\{1 + \exp(\mathbf{z}'\boldsymbol{\gamma})\}$
`probit`	$\Phi(\mathbf{z}'\boldsymbol{\gamma})$

3. Both $\mathbf{x}$ and $\mathbf{z}$ can be high dimensional, and there may be more variables than the number of observations. However, the outcome model parameter $\boldsymbol{\beta}_t$ and the treatment model parameter $\boldsymbol{\gamma}$ are assumed to be sparse. This means that there are only a few nonzero elements in both $\boldsymbol{\beta}_t$ and $\boldsymbol{\gamma}$.
4. Although we assume that the functional forms of $g_0(\cdot)$ and $m_0(\cdot)$ are known, we do not know which variables should enter the model. We use lasso techniques to select variables from the potential high-dimensional controls $\mathbf{x}$ and $\mathbf{z}$. The resulting estimators should guard against possible model selection errors made by lassos.

Next we discuss the methods that `telasso` uses to estimate ATEs, ATETs, and POMs.

Neyman orthogonal moments

Denote θ_0 as the parameter of interest, which can be ATE, ATET, or POM. Following Chernozhukov et al. (2018), we use the following moment conditions to estimate θ_0.

The moment condition for estimating ATE is

$$\psi(\mathbf{w};\theta,\boldsymbol{\eta}) = \{g(1,\mathbf{x}) - g(0,\mathbf{x})\} + \frac{\tau\,\{y - g(1,\mathbf{x})\}}{m(\mathbf{z})} - \frac{(1-\tau)\,\{y - g(0,\mathbf{x})\}}{1-m(\mathbf{z})} - \theta \tag{1}$$

where $\mathbf{w} = (y, \mathbf{x}', \mathbf{z}', \tau)'$ and $\boldsymbol{\eta}$ is the nuisance parameter consisting of $g(\tau,\mathbf{x})$ and $m(\mathbf{z})$. Using the variables selected by lassos, $g(1,\mathbf{x})$, $g(0,\mathbf{x})$, and $m(\mathbf{z})$ are approximations to the true functions $g_0(1,\mathbf{x})$, $g_0(0,\mathbf{x})$, and $m_0(\mathbf{z})$, respectively.

The moment condition for estimating ATET is

$$\psi(\mathbf{w};\theta;\boldsymbol{\eta}) = \frac{\tau\{y - g(0,\mathbf{x})\}}{p} - \frac{m(\mathbf{z})(1-\tau)\{y - g(0,\mathbf{x})\}}{p\{1-m(\mathbf{z})\}} - \frac{\tau\theta}{p} \tag{2}$$

where $p = E(\tau)$.

The moment condition for estimating POM for $\tau = 1$ is

$$\psi(\mathbf{w};\theta,\boldsymbol{\eta}) = g(1,\mathbf{x}) + \frac{\tau\,\{y - g(1,\mathbf{x})\}}{m(\mathbf{z})} - \theta \tag{3}$$

The moment condition for estimating POM for $\tau = 0$ is

$$\psi(\mathbf{w};\theta,\boldsymbol{\eta}) = g(0,\mathbf{x}) + \frac{(1-\tau)\,\{y - g(0,\mathbf{x})\}}{1-m(\mathbf{z})} - \theta \tag{4}$$

To estimate θ_0, we first use lasso techniques to select variables in the outcome and treatment models. Based on the selected variables, we can estimate the post-lasso predictions for $g(\tau,\mathbf{x})$ and $m(\mathbf{z})$. Denote $\widetilde{\boldsymbol{\eta}}$ as the estimates for $g(\tau,\mathbf{x})$ and $m(\mathbf{z})$.

The estimator $\widehat{\theta}$ for θ_0 is the solution to

$$\frac{1}{n}\sum_{i=1}^{n}[\psi_i(\mathbf{w};\theta,\widetilde{\boldsymbol{\eta}})] = 0 \tag{5}$$

The variance estimator for $\widehat{\theta}$ is

$$\frac{1}{n^2}\sum_{i=1}^{n}\left[\psi_i(\mathbf{w};\widehat{\theta},\widetilde{\boldsymbol{\eta}})^2\right] = 0$$

To identify θ_0, we need to assume the conditional mean independence, overlap, and independent treatment assignment. For a detailed discussion of these assumptions, see [CAUSAL] **teffects intro** or [CAUSAL] **teffects intro advanced**.

The moment conditions defined in (1) to (4) are Neyman orthogonal. Intuitively, Neyman orthogonal moments mean that the estimator or parameter of interest is still consistent even if model selection makes some mild mistakes. For a formal introduction on Neyman orthogonality, see section 2.1 in Chernozhukov et al. (2018).

The moment conditions in (1) to (4) also imply the AIPW estimator for treatment effects. The AIPW estimator is doubly robust in the sense that only one of the outcome model or the treatment model is required to be correctly specified.

Double machine learning

`telasso` also implements the double machine learning estimator in Chernozhukov et al. (2018) for ATEs, ATETs, and POMs. One advantage of double machine learning is that it allows `telasso` to select more variables in the models. The sparsity assumption is crucial for the validity of the estimators implemented in `telasso`. In other words, the sparsity assumption allows only a few variables to be selected. Double machine learning techniques relax this sparsity requirement to some extent such that more variables can be selected and the estimation results are still valid.

`telasso` with the `xfold()` option implements cross-fit estimation for ATEs, ATETs, and POMs. Cross-fit estimation fits the nuisance parameters and parameter of interest in different samples. It has the following structure.

Let K be the specified number of cross-fitting folds.

1. Randomly partition the sample into K subsamples called folds.
2. Define I_k to be the observations in fold k, and define IC_k to be the sample observations not in fold k.
3. For each $k = 1, \ldots, K$, fill in the observations of $i \in I_k$ for the moment condition $\psi(\cdot)$ in (1), (2), (3), or (4) depending on the specified statistic.
   a. Using observations $i \in IC_k$, perform model selection, and estimate the parameters in the outcome and treatment models.
   b. Based on the estimates in step 3a, predict the nuisance parameter $\widetilde{\boldsymbol{\eta}}_k$ for observations $i \in I_k$.
   c. Based on $\widetilde{\boldsymbol{\eta}}_k$, fill in the moment conditions $\psi(\cdot)$ for observations $i \in I_k$.
4. The estimator $\widehat{\theta}$ is the solution in (5).

Resampling the partitions

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

Let S be the specified number of resamples.

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

$$\widetilde{\theta} = \frac{1}{S}\sum_{s=1}^{S}\widehat{\theta}_s$$

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

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

References

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

Chernozhukov, V., and C. B. Hansen. 2004. The effects of 401(k) participation on the wealth distribution: An instrumental quantile regression analysis. *Review of Economics and Statistics* 86: 735–751. https://doi.org/10.1162/0034653041811734.

Farrell, M. H. 2015. Robust inference on average treatment effects with possibly more covariates than observations. *Journal of Econometrics* 189: 1–23. https://doi.org/10.1016/j.jeconom.2015.06.017.

Koch, B., D. M. Vock, and J. Wolfson. 2018. Covariate selection with group lasso and doubly robust estimation of causal effects. *Biometrics* 74: 8–17. https://doi.org/10.1111/biom.12736.

Leeb, H., and B. M. Pötscher. 2005. Model selection and inference: Facts and fiction. *Econometric Theory* 21: 21–59. https://doi.org/10.1017/S0266466605050036.

——. 2006. Can one estimate the conditional distribution of post-model-selection estimators? *Annals of Statistics* 34: 2554–2591. https://doi.org/10.1214/009053606000000821.

Poterba, J. M., S. F. Venti, and D. A. Wise. 1994. 401(k) plans and tax-deferred saving. In *Studies in the Economics of Aging*, ed. D. A. Wise, 105–142. New York: National Bureau of Economic Research.

——. 1995. Do 401(k) contributions crowd out other personal saving? *Journal of Public Economics* 58: 1–32. https://doi.org/10.1016/0047-2727(94)01462-W.

Rosenbaum, P. R., and D. B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70: 41–55. https://doi.org/10.2307/2335942.

Also see

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

[CAUSAL] **teffects aipw** — Augmented inverse-probability weighting

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

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