Statistics/Data Analysis

help binstest

Title

binstest —— Data-Driven Nonparametric Shape Restriction and Parametric Model Specification Testing using Binscatter.

Syntax

```
binstest depvar indvar [othercovs] [if] [in] [weight] [ ,
        estmethod(cmdname) deriv(v) at(position) nolink
        absorb(absvars) reghdfeopt(reghdfe_option)
        testmodel(testmodelopt) testmodelparfit(filename) testmodelpoly(p)
        testshape(testshapeopt) testshapel(numlist) testshaper(numlist)
        testshape2(numlist) lp(metric)
        bins(p s) nbins(nbinsopt) binspos(position) binsmethod(method)
        nbinsrot(#) randcut(#)
        pselect(numlist) sselect(numlist)
        nsims(#) simsgrid(#) simsseed(seed)
        dfcheck(n1 n2) masspoints(masspointsoption)
        vce(vcetype) asyvar(on/off) estmethodopt(cmd_option) usegtools(on/off) ]
```

where *depvar* is the dependent variable, *indvar* is the independent variable for binning, and *othercovs* are other covariates to be controlled for.

The degree of the piecewise polynomial p, the number of smoothness constraints s, and the derivative order v are integers satisfying 0 <= s,v <= p, which can take different values in each case.

At least one test has to be specified via **testmodelparfit()**, **testmodelpoly()**, **testshapel()**, **testshaper()** and/or **testshape2()**.

fweights, **aweight**s and **pweight**s are allowed; see weight.

Description

binstest implements binscatter-based hypothesis testing procedures for parametric functional forms of and nonparametric shape restrictions on the regression function estimators, following the results in Cattaneo, Crump, Farrell and Feng (2023a) and Cattaneo, Crump, Farrell and Feng (2023b). If the binning scheme is not set by the user, the companion command binsregselect is used to implement binscatter in a data-driven (optimal) way and inference procedures are based on robust bias correction. Binned scatter plots based on different models can be constructed using the companion commands binsreg, binsqreg, binslogit and binsprobit.

A detailed introduction to this command is given in Cattaneo, Crump, Farrell and Feng (2023c). Companion R and Python packages with the same capabilities are available (see website below).

Companion commands: binsreg for binscatter regression with robust inference procedures and plots, binsqreg for binscatter quantile regression with robust inference procedures and plots, binslogit for binscatter logit estimation with robust inference procedures and plots, binsprobit for binscatter probit estimation with robust inference procedures and plots, and binsregselect for data-driven (optimal) binning selection.

Related Stata, R and Python packages are available in the following website:

https://nppackages.github.io/

Options

Estimand

estmethod(*cmdname***)** specifies the binscatter model. The default is **estmethod(reg)**, which corresponds to the binscatter least squares regression. Other options are: **estmethod(qreg #)** for binscatter quantile regression where # is the quantile to be estimated, **estmethod(logit)** for binscatter logistic regression and **estmethod(probit)** for binscatter probit regression.

deriv(*v***)** specifies the derivative order of the regression function for estimation, testing and plotting. The default is **deriv(0)**, which corresponds to the function itself.

at(*position***)** specifies the values of *othercovs* at which the estimated function is evaluated for plotting. The default is **at(mean)**, which corresponds to the mean of *othercovs*. Other options are: **at(median)** for the median of *othercovs*, **at(0)** for zeros, and **at(filename)** for particular values of *othercovs* saved in another file.

Note: When **at(mean)** or **at(median)** is specified, all factor variables in *othercovs* (if specified) are excluded from the evaluation (set as zero).

nolink specifies that the function within the inverse link (logistic) function be reported instead of the conditional probability function. This option is used only if logit or probit model is specified in **estmethod()**.

Reghdfe

absorb(*absvars***)** specifies categorical variables (or interactions) representing the fixed effects to be absorbed. This is equivalent to including an indicator/dummy variable for each category of each *absvar*. When **absorb()** is specified, the community-contributed command **reghdfe** instead of the command **regress** is used.

reghdfeopt(*reghdfe_option***)** options to be passed on to the command **reghdfe**. Important: **absorb()** and **vce()** should not be specified within this option.

For more information about the community-contributed command **reghdfe**, please see http://scorreia.com/software/reghdfe/.

Parametric Model Specification Testing

testmodel(*testmodelopt***)** sets the degree of polynomial and the number of smoothness constraints for parametric model specification testing. If **testmodel(p s)** is specified, a piecewise polynomial of degree *p* with *s* smoothness constraints is used. If **testmodel(T)** or **testmodel()** is specified, **testmodel(1 1)** is used unless the degree *p* or smoothness *s* selection is requested via the option **pselect()** or **sselect()** (see more details in the explanation of **pselect()** and **sselect()**). The default is **testmodel()**.

testmodelparfit(*filename***)** specifies a dataset which contains the evaluation grid and fitted values of the model(s) to be tested against. The file must have a variable with the same name as *indvar*, which contains a series of evaluation points at which the binscatter model and the parametric model of interest are compared with each other. Each parametric model is represented by a variable named as *binsreg_fit**, which must contain the fitted values at the corresponding evaluation points.

testmodelpoly(*p***)** specifies the degree of a global polynomial model to be tested against.

Nonparametric Shape Restriction Testing

testshape(*testshapeopt***)** sets the degree of polynomial and the number of smoothness constraints for nonparametric shape restriction testing. If **testshape(p s)** is specified, a piecewise polynomial of degree *p* with *s* smoothness constraints is used. If **testshape(T)** or **testshape()** is specified, **testshape(1 1)** is used unless the degree *p* or smoothness *s* selection is requested via the option **pselect()** or **sselect()** (see more details in the explanation of **pselect()** and **sselect()**). The default is **testshape()**.

testshapel(*numlist***)** specifies a numlist of null boundary values for hypothesis testing. Each number *a* in the *numlist* corresponds to one boundary of a one-sided hypothesis test to the left of the form H0: *sup_x mu(x)<=a*.

testshaper(*numlist***)** specifies a numlist of null boundary values for hypothesis testing. Each number *a* in the *numlist* corresponds to one boundary of a one-sided hypothesis test to the right of the form H0: *inf_x mu(x)>=a*.

testshape2(*numlist***)** specifies a numlist of null boundary values for hypothesis testing. Each number *a* in the *numlist* corresponds to one boundary of a two-sided hypothesis test of the form H0: *sup_x |mu(x)-a|=0*.

Metric for Hypothesis Testing

lp(*metric***)** specifies an Lp metric used for (two-sided) parametric model specification testing and/or shape restriction testing. The default is **lp(inf)**, which corresponds to the sup-norm. Other options are **lp(q)** for a positive integer **q**.

Binning/Degree/Smoothness Selection

bins(*p s***)** sets a piecewise polynomial of degree *p* with *s* smoothness constraints for data-driven (IMSE-optimal) selection of the partitioning/binning scheme. The default is **bins(0 0)**, which corresponds to the piecewise constant.

nbins(*nbinsopt***)** sets the number of bins for partitioning/binning of *indvar*. If **nbins(T)** or **nbins()** (default) is specified, the number of bins is selected via the companion command binsregselect in a data-driven, optimal way whenever possible. If a numlist with more than one number is specified, the number of bins is selected within this list via the companion command binsregselect.

binspos(*position***)** specifies the position of binning knots. The default is **binspos(qs)**, which corresponds to quantile-spaced binning (canonical binscatter). Other options are: **es** for evenly-spaced binning, or a numlist for manual specification of the positions of inner knots (which must be within the range of *indvar*).

binsmethod(*method***)** specifies the method for data-driven selection of the number of bins via the companion command binsregselect. The default is **binsmethod(dpi)**, which corresponds to the IMSE-optimal direct plug-in rule. The other option is: **rot** for rule of thumb implementation.

nbinsrot(*#***)** specifies an initial number of bins value used to construct the DPI number of bins selector. If not specified, the data-driven ROT selector is used instead.

randcut(*#***)** specifies the upper bound on a uniformly distributed variable used to draw a subsample for bins/degree/smoothness selection. Observations for which **runiform()<=#** are used. # must be between 0 and 1. By default, max(5000, 0.01n) observations are used if the samples size n>5000.

pselect(*numlist***)** specifies a list of numbers within which the degree of polynomial *p* for point estimation is selected. If the selected optimal degree is *p*, then piecewise polynomials of degree *p+1* are used to conduct testing for nonparametric shape restrictions or parametric model specifications.

sselect(*numlist***)** specifies a list of numbers within which the number of smoothness constraints *s* for point estimation. If the selected optimal smoothness is *s*, then piecewise polynomials with *s+1* smoothness constraints are used to conduct testing for nonparametric shape restrictions or parametric model specifications. If not specified, for each value *p* supplied in the option **pselect()**, only the piecewise polynomial with the maximum smoothness is considered, i.e., *s=p*.

Note: To implement the degree or smoothness selection, in addition to **pselect()** or **sselect()**, **nbins(#)** must be specified.

Simulation

nsims(#**)** specifies the number of random draws for hypothesis testing. The default is **nsims(500)**, which corresponds to 500 draws from a standard Gaussian random vector of size [(p+1)*J - (J-1)*s]. Setting at least **nsims(2000)** is recommended to obtain the final results.

simsgrid(#**)** specifies the number of evaluation points of an evenly-spaced grid within each bin used for evaluation of the supremum (infimum or Lp metric) operation needed for hypothesis testing procedures. The default is **simsgrid(20)**, which corresponds to 20 evenly-spaced evaluation points within each bin for approximating the supremum (infimum or Lp metric) operator. Setting at least **simsgrid(50)** is recommended to obtain the final results.

simsseed(#**)** sets the seed for simulations.

Mass Points and Degrees of Freedom

dfcheck(*n1 n2***)** sets cutoff values for minimum effective sample size checks, which take into account the number of unique values of *indvar* (i.e., adjusting for the number of mass points), number of clusters, and degrees of freedom of the different statistical models considered. The default is **dfcheck(20 30)**. See Cattaneo, Crump, Farrell and Feng (2023c) for more details.

masspoints(*masspointsoption***)** specifies how mass points in *indvar* are handled. By default, all mass point and degrees of freedom checks are implemented. Available options:
masspoints(*noadjust***)** omits mass point checks and the corresponding effective sample size adjustments.
masspoints(*nolocalcheck***)** omits within-bin mass point and degrees of freedom checks.
masspoints(*off***)** sets **masspoints(***noadjust***)** and **masspoints(***nolocalcheck***)** simultaneously.
masspoints(*veryfew***)** forces the command to proceed as if *indvar* has only a few number of mass points (i.e., distinct values). In other words, forces the command to proceed as if the mass point and degrees of freedom checks were failed.

Other Options

vce(*vcetype***)** specifies the *vcetype* for variance estimation used by the commands regress, logit, probit, qreg or **reghdfe**. The default is **vce(robust)**.

asyvar(*on/off***)** specifies the method used to compute standard errors. If **asyvar(on)** is specified, the standard error of the nonparametric component is used and the uncertainty related to other control variables *othercovs* is omitted. Default is **asyvar(off)**, that is, the uncertainty related to *othercovs* is taken into account.

estmethodopt(*cmd_option***)** options to be passed on to the estimation command specified in **estmethod()**. For example, options that control for the optimization process can be added here.

usegtools(*on/off***)** forces the use of several commands in the community-distributed Stata package **gtools** to speed the computation up, if *on* is specified. Default is **usegtools(off)**.

For more information about the package **gtools**, please see https://gtools.readthedocs.io/en/latest/index.html.

Examples

Setup
```
. sysuse auto
```

Test for linearity
```
. binstest mpg weight foreign, testmodelpoly(1)
```

Test for monotonicity
```
. binstest mpg weight foreign, deriv(1) bins(1 1) testshapel(0)
```

Stored results

Scalars

e(N)	number of observations
e(Ndist)	number of distinct values
e(Nclust)	number of clusters
e(nbins)	number of bins
e(p)	degree of polynomial for bin selection
e(s)	smoothness of polynomial for bin selection
e(testshape_p)	degree of polynomial for testing shape restrictions
e(testshape_s)	smoothness of polynomial for testing shape restrictions
e(testmodel_p)	degree of polynomial for testing model specifications
e(testmodel_s)	smoothness of polynomial for testing model specifications
e(testpolyp)	degree of polynomial regression model
e(stat_poly)	statistic for testing global polynomial model
e(pval_poly)	p value for testing global polynomial model
e(imse_var_rot)	variance constant in IMSE, ROT selection
e(imse_bsq_rot)	bias constant in IMSE, ROT selection
e(imse_var_dpi)	variance constant in IMSE, DPI selection
e(imse_bsq_dpi)	bias constant in IMSE, DPI selection

Macros

e(testvarlist)	varlist found in **testmodel()**
e(testvalue2)	values in **testshape2()**
e(testvalueR)	values in **testshaper()**
e(testvalueL)	values in **testshapel()**

Matrices

e(pval_model)	p values for **testmodel()**
e(stat_model)	statistics for **testmodel()**
e(pval_shape2)	p values for **testshape2()**
e(stat_shape2)	statistics for **testshape2()**
e(pval_shapeR)	p values for **testshaper()**
e(stat_shapeR)	statistics for **testshaper()**
e(pval_shapeL)	p values for **testshapel()**
e(stat_shapeL)	statistics for **testshapel()**

References

Cattaneo, M. D., R. K. Crump, M. H. Farrell, and Y. Feng. 2023a. On Binscatter. Working Paper.

Cattaneo, M. D., R. K. Crump, M. H. Farrell, and Y. Feng. 2023b. Nonlinear Binscatter Methods. Working Paper.

Cattaneo, M. D., R. K. Crump, M. H. Farrell, and Y. Feng. 2023c. Binscatter Regressions. Working Paper.

Authors

Matias D. Cattaneo, Princeton University, Princeton, NJ. cattaneo@princeton.edu.

Richard K. Crump, Federal Reserve Band of New York, New York, NY. richard.crump@ny.frb.org.

Max H. Farrell, UC Santa Barbara, Santa Barbara, CA. mhfarrell@gmail.com.

Yingjie Feng, Tsinghua University, Beijing, China. fengyingjiepku@gmail.com.