npregress series — Nonparametric series regression

Description

`npregress series` performs nonparametric series estimation using a B-spline, piecewise polynomial spline, or polynomial basis. Like linear regression, nonparametric regression models the mean of the outcome conditional on the covariates, but unlike linear regression, it makes no assumptions about the functional form of the relationship between the outcome and the covariates. `npregress series` may be used to model the mean of a continuous, count, or binary outcome.

Quick start

Nonparametric regression of `y` on `x` and discrete covariate `a` using the default B-spline basis

```
npregress series y x i.a
```

Same as above, but use a polynomial basis

```
npregress series y x i.a, polynomial
```

Same as above, but use a piecewise polynomial spline basis

```
npregress series y x i.a, spline
```

Same as above, but use AIC to find the optimal basis function

```
npregress series y x i.a, criterion(aic) spline
```

Interpolate using three knots

```
npregress series y x i.a, knots(3)
```

Specify values of knots in matrix `K`

```
npregress series y x i.a, knotsmat(K)
```

Menu

Statistics > Nonparametric analysis > Nonparametric series regression

Syntax

`npregress series` *depvar* *indepvars*$_{\text{series}}$ [*if*] [*in*] [*weight*] [, *options*]

indepvars$_{\text{series}}$ is the list of independent variables for which a basis function will be formed.

options	Description
Model	
`bspline`	use a third-order B-spline basis; the default
`bspline(#)`	use a B-spline basis of order #
`spline`	use a third-order piecewise polynomial spline basis
`spline(#)`	use a piecewise polynomial spline basis of order #
`polynomial`	use a polynomial basis
`polynomial(#)`	use a polynomial basis of order #
`asis(`*varlist*`)`	include *varlist* in model as specified; do not use in basis
`nointeract(`*seriesvarlist*`)`	use *seriesvarlist* in basis without interactions
`criterion(`*crittype*`)`	criterion to use; *crittype* may be `cv`, `gcv`, `aic`, `bic`, or `mallows`
`knots(#)`	use a piecewise polynomial spline or B-spline basis function with # knots
`knotsmat(`*matname*`)`	use knots in matrix *matname* for piecewise polynomial spline or B-spline estimation
`distinct(#)`	minimum number of distinct values allowed in continuous covariates; default is `distinct(10)`
`basis(`*stub* [`, replace`]`)`	store elements of piecewise polynomial spline or B-spline basis function using *stub*
`rescale(`*stub* [`, replace`]`)`	store rescaled values of covariates using *stub*
SE	
`vce(`*vcetype*`)`	*vcetype* may be `robust`, `ols`, or `bootstrap`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`aequations`	display auxiliary regression coefficients
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process
`coeflegend`	display legend instead of statistics

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

`bootstrap`, `by`, `collect`, and `jackknife` are allowed; see **[U] 11.1.10 Prefix commands**.

Weights are not allowed with the `bootstrap` prefix; see **[R] bootstrap**.

`fweight`s and `iweight`s are allowed; see **[U] 11.1.6 weight**.

`coeflegend` does not appear in the dialog box.

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

Options

Model

`bspline` specifies that a third-order B-spline be selected. It is the default basis.

`bspline(`*#*`)` specifies that a B-spline of order # be used as the basis. The order may be 1, 2, or 3.

`spline` specifies that a third-order piecewise polynomial spline be selected as the basis.

`spline(`*#*`)` specifies that a piecewise polynomial spline of order # be used as the basis. The order may be 1, 2, or 3.

`polynomial` specifies that a polynomial be selected as the basis.

`polynomial(`*#*`)` specifies that a polynomial of order # be used as the basis. The order may be an integer between 1 and 16.

`asis(`*varlist*`)` specifies that variables in *varlist* be included as independent variables in the model without any transformation. No B-spline, piecewise polynomial spline, or polynomial basis function will be formed from these variables. Variables in *varlist* may not be specified in *indepvars*$_{\text{series}}$.

`nointeract(`*seriesvarlist*`)` specifies that the terms in the basis function formed from variables in *seriesvarlist* not be interacted with the terms of the basis function formed from other variables in *indepvars*$_{\text{series}}$. Covariates specified in *seriesvarlist* must be in *indepvars*$_{\text{series}}$.

`criterion(`*crittype*`)` specifies that *crittype* be used to select the optimal number of terms in the basis function. *crittype* may be one of the following: `cv` (cross-validation), `gcv` (generalized cross-validation), `aic` (Akaike's information criterion), `bic` (Schwarz's Bayesian information criterion), or `mallows` (Mallows's C_p). The default is `criterion(cv)`.

`knots(`*#*`)` specifies that a piecewise polynomial spline or B-spline basis function with # knots be used. The minimum number of knots must be an integer greater than or equal to 1. The maximum number of knots is either 4,096 or two-thirds of the sample size, whichever is smaller.

`knotsmat(`*matname*`)` specifies that the knots for each continuous covariate be the values in each row of *matname*. The number of knots should be the same for each covariate, and there must be as many rows as there are continuous covariates. If rows of *matname* are not labeled with *varname*s, then rows are assumed to be in the order of *indepvars*$_{\text{series}}$.

`distinct(`*#*`)` specifies the minimum number of distinct values allowed in continuous variables. By default, continuous variables that enter the basis through either *indepvars*$_{\text{series}}$ or *seriesvarlist* are required to have at least 10 distinct values. Continuous variables with few distinct values provide little information for determining an appropriate basis function and may produce unreliable estimates.

`basis(`*stub* [`, replace`]`)` specifies that the elements of the basis function generated by `npregress series` be stored with the specified names.

The option argument *stub* is the prefix used to generate enumerated variables for each element of the basis.

When `replace` is used, existing variables named with *stub* are replaced by those from the new computation.

`rescale(`*stub* [`, replace`]`)` specifies that the rescaled covariates used to generate the basis function be stored with the specified names.

The option argument *stub* is the prefix used to generate enumerated variable names for the covariates.

When `replace` is used, existing covariates named with *stub* are replaced by those from the new computation.

SE

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification (`robust`), that assume homoskedasticity (`ols`), and that use bootstrap methods (`bootstrap`); see [R] ***vce_option***.

Reporting

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

`aequations` specifies that the auxiliary regression coefficients be reported. By default, only the average marginal effects of the covariates on the outcome are reported.

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**.

Maximization

maximize_options: `iterate(`*#*`)`, [`no`]`log`, `tolerance(`*#*`)`; see [R] **Maximize**. These options are seldom used.

The following option is available with `npregress series` but is not shown in the dialog box:

`coeflegend`; see [R] **Estimation options**.

Remarks and examples

stata.com

This entry assumes that you are already familiar with nonparametric regression. Below, we discuss nonparametric series estimation; see [R] **npregress intro** for an overview of nonparametric regression and the models fit by `npregress series` and `npregress kernel`.

Remarks are presented under the following headings:

Overview

`npregress series` implements nonparametric series estimation using a B-spline, piecewise polynomial spline, or polynomial basis. The covariates may be continuous or discrete. `npregress series` allows you to estimate covariate effects and other counterfactuals related to the unknown mean function after estimation.

The word “nonparametric” refers to the fact that the parameter of interest—the mean as a function of the covariates—is given by the unknown function $g(\mathbf{x}_i)$, which is an element of an infinite-dimensional space of functions. In contrast, in a parametric model, the mean for a given value of the covariates, $E(y_i|\mathbf{x}_i) = f(\mathbf{x}_i, \boldsymbol{\beta})$, is a known function that is fully characterized by the parameter of interest, $\boldsymbol{\beta}$, which is a finite-dimensional real vector (Shao 2003).

The nonparametric regression model of outcome y_i given the k-dimensional vector of covariates $\mathbf{x}_i$ is given by

$$y_i = g\left(\mathbf{x}_i\right) + \varepsilon_i \tag{1}$$

$$E\left(\varepsilon_i|\mathbf{x}_i\right) = 0 \tag{2}$$

where ε_i is the error term. Equations (1) and (2) imply that

$$E\left(y_i|\mathbf{x}_i\right) = g\left(\mathbf{x}_i\right)$$

Once we account for the information in the covariates, the error term provides no information about the mean of our outcome. The conditional mean function is therefore given by $g(\mathbf{x}_i)$.

The mean estimate we obtain using nonparametric series estimation has the same form of the mean function estimate we obtain using linear regression. The regressors, however, are not variables in the data but functions of the variables. An example would be a kth-order polynomial. Suppose we have one covariate. The elements of the polynomial in this case would be $\left(x_i, x_i^2, \ldots, x_i^k\right)$. If we define $\mathbf{z}_i$ as a vector with elements $\left(x_i, x_i^2, \ldots, x_i^k\right)$, we may write the estimate of the mean function as

$$\mathbf{z}_i'\widehat{\boldsymbol{\beta}}$$

where $\widehat{\boldsymbol{\beta}}$ has the form of an ordinary least-squares estimate.

`npregress series` allows us to specify other functional forms for $\mathbf{z}_i$ depending on the basis we select: B-spline, piecewise polynomial spline, or polynomial. See *Methods and formulas* for the formulas for each basis.

Although the estimate of the mean function has the form of a linear regression, the individual coefficients are not easily interpretable. For instance, in our kth-order polynomial example, if x_i is continuous, the marginal effect of x_i is not a single coefficient but rather is a function of k elements of $\boldsymbol{\beta}$ and the covariate x_i.

In the example above, we had only one covariate, x_i. If we have more than one covariate, we approximate the mean function by using interactions of the terms in the basis function for each covariate. For instance, a polynomial of x_i and w_i would have terms $\left(x_i, w_i, x_i w_i, x_i^2, w_i^2, \ldots, w_i^k x_i^k\right)$. As the number of covariates increases, the number of terms in the basis function increases exponentially. This is referred to in the literature as the curse of dimensionality.

`npregress series` allows us to reduce the dimensionality by using the `nointeract()` option to request that some covariates not be interacted with others. For the example above, this is equivalent to specifying a model of the form

$$y_i = g_1\left(x_i\right) + g_2\left(w_i\right) + \varepsilon_i \tag{3}$$

In (3), $g_1\left(x_i\right)$ and $g_2\left(w_i\right)$ are unknown functions, but there are no interactions between x_i and w_i. This ameliorates the curse of dimensionality but imposes more structure to the model.

You may also want to reduce the curse of dimensionality by requesting a parametric component, by using the `asis()` option, to fit models like this:

$$y_i = g\left(x_i\right) + w_i\boldsymbol{\beta} + \varepsilon_i \tag{4}$$

In (4), $g\left(x_i\right)$ is unknown but we assume that w_i enters the model linearly.

As mentioned above, the regression coefficients are not easily interpretable. We can, however, estimate marginal effects, as reported in the `npregress series` output, and use `margins` to answer specific questions about the effects of covariates on the conditional mean, $g\left(\mathbf{x}_i\right)$. We demonstrate this in the examples below.

For detailed introductions to series estimators and the methods implemented by `npregress series`, see de Boor (2001), Schumaker (2007), Eubank (1999), Schoenberg (1969), Newey (1997), and Chen (2007).

Estimation and effects

▷ Example 1: Nonparametric series regression estimation

dui.dta contains information about the number of monthly drunk driving citations in a local jurisdiction (citations). Suppose we want to know the effect of increasing fines on the number of citations. Because citations is a count variable, we could consider fitting the model with poisson or nbreg. However, both of these estimators make assumptions about the distribution of the data. If these assumptions are not true, we will obtain inconsistent estimates.

By using npregress series, we do not have to make any assumptions about how citations is distributed. We use npregress series to estimate the average marginal effect of drunk driving penalties (fines) on citations.

```
. use https://www.stata-press.com/data/r18/dui
(Fictional data on monthly drunk driving citations)

. npregress series citations fines

Computing approximating function

Minimizing cross-validation criterion

Iteration 0:  Cross-validation criterion =  55.15697
Iteration 1:  Cross-validation criterion =  55.11413

Computing average derivatives

Cubic B-spline estimation                   Number of obs     =         500
Criterion: cross-validation                 Number of knots   =           3

------------------------------------------------------------------------------
             |               Robust
   citations |     Effect   std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
       fines |  -8.020769    .464836   -17.26   0.000    -8.931831   -7.109707
------------------------------------------------------------------------------
Note: Effect estimates are averages of derivatives.
```

The iteration log first tells us that the approximating function is being computed. At this stage, the number of knots of the cubic B-spline is selected using cross-validation. Three knots were selected.

After the approximating function is computed, average marginal effects are computed. This second step is computationally expensive. The computation time increases with the number of elements in the basis function, which in turn increases with the complexity of the mean function we are trying to compute.

The table reports that the average marginal effect of fines on the mean number of citations is −8.02. Increasing fines, on average, reduces the number of citations.

◁

npregress series generates a system variable for each element of the basis function. Additionally, variables are generated with the rescaled values of the continuous covariates used to construct the basis function. To see the variables that npregress series generated for example 1, we type

```
. describe *_*, fullnames

Variable      Storage   Display    Value
    name         type    format    label      Variable label
--------------------------------------------------------------------------------

__x1rs          double  %10.0g                fines rescaled to [0,1]
_x1__b1         double  %10.0g                Basis term 1 for fines
_x1__b2         double  %10.0g                Basis term 2 for fines
_x1__b3         double  %10.0g                Basis term 3 for fines
_x1__b4         double  %10.0g                Basis term 4 for fines
_x1__b5         double  %10.0g                Basis term 5 for fines
_x1__b6         double  %10.0g                Basis term 6 for fines
_x1__b7         double  %10.0g                Basis term 7 for fines
```

To specify a name for each of the elements of the basis function, we can use the basis() option with a *stub*.

```
. npregress series citations fines, basis(basis)
  (output omitted )
```

We get the following set of names for the elements of the basis function:

```
. describe basis*, fullnames

Variable      Storage   Display    Value
    name         type    format    label      Variable label
--------------------------------------------------------------------------------

basis1          double  %10.0g                Basis term 1 for fines
basis2          double  %10.0g                Basis term 2 for fines
basis3          double  %10.0g                Basis term 3 for fines
basis4          double  %10.0g                Basis term 4 for fines
basis5          double  %10.0g                Basis term 5 for fines
basis6          double  %10.0g                Basis term 6 for fines
basis7          double  %10.0g                Basis term 7 for fines
```

We may also modify the name of the rescaled variable by using the rescale() option.

```
. npregress series citations fines, rescale(rescaled)
  (output omitted )
```

This will give us

```
. describe rescaled*, fullnames

              storage   display    value
variable name   type    format     label      variable label
--------------------------------------------------------------------------------

rescaled1       double  %10.0g                fines rescaled to [0,1]
```

▷ Example 2: Estimation with more than one regressor

We now extend example 1. In addition to fines, we model citations as a function of whether the jurisdiction is small, medium, or large (csize) and whether there is a college in the jurisdiction (college).

```
. npregress series citations fines i.csize i.college
Computing approximating function
Minimizing cross-validation criterion
Iteration 0:   Cross-validation criterion =  30.26251
Computing average derivatives
Cubic B-spline estimation                    Number of obs      =          500
Criterion: cross-validation                  Number of knots    =            1

------------------------------------------------------------------------------
             |               Robust
   citations |     Effect   std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
       fines |  -7.787386   .2917941   -26.69   0.000    -8.359292   -7.215481
             |
       csize |
     (Medium |
          vs |
      Small) |   4.732592   .5087968     9.30   0.000     3.735368    5.729815
      (Large |
          vs |
      Small) |   10.91757   .5350892    20.40   0.000     9.868813    11.96632
             |
     college |
    (College |
          vs |
  Not coll..) |   6.514286   .5958949    10.93   0.000     5.346353    7.682218
------------------------------------------------------------------------------
Note: Effect estimates are averages of derivatives for continuous covariates
      and averages of contrasts for factor covariates.
```

The average marginal effect of fines is −7.79, slightly less in magnitude than the −8.02 that we estimated in example 1. The output also shows effects for the variables csize and college. In these categorical variables, the effects are differences instead of derivatives. For example, if every jurisdiction in the population were a college town, we would expect 6.51 more citations than if none were college towns.

◁

▷ Example 3: Expected citations for different levels of fines

The npregress series command reported that the average marginal effect of fines on number of citations is negative. We can use margins to further explore the relationship between level of fines and expected number of citations. What would we expect if all jurisdictions set fines to $8,000? What if they all set fines to $9,000? $10,000? $11,000? We use the at(fines=(8 9 10 11)) option with margins to estimate these expected values.

```
. margins, at(fines=(8 9 10 11))

Predictive margins                                    Number of obs = 500
Model VCE: Robust

Expression: Mean function, predict()
1._at: fines =  8
2._at: fines =  9
3._at: fines = 10
4._at: fines = 11

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         _at |
          1  |   49.58234    1.47392    33.64   0.000     46.69351    52.47117
          2  |   28.35154   .5730302    49.48   0.000     27.22842    29.47466
          3  |   20.40163   .3320855    61.43   0.000     19.75075     21.0525
          4  |   14.78085   .4297201    34.40   0.000     13.93862    15.62309
------------------------------------------------------------------------------
```

There appears to be a dramatic drop in the expected number of citations as fines increase from $8,000 to $9,000. We can visualize these results if we type `marginsplot`.

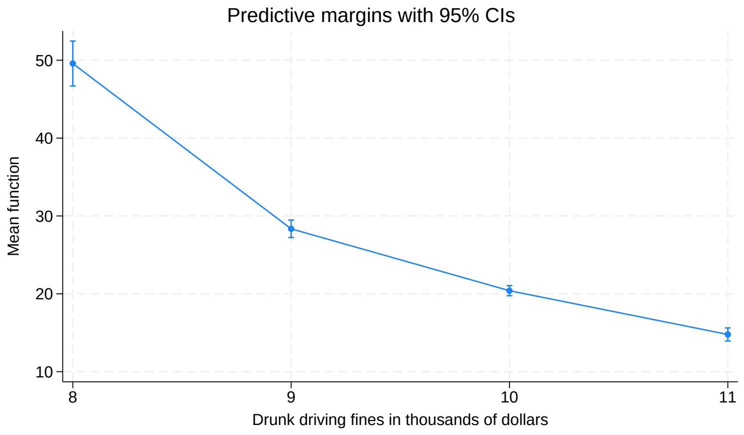

Figure 1.

Are there significant differences in the expected number of citations as we increase fines in increments of $1,000? If we use the reverse-adjacent contrast operator, `ar.`, with `margins`, we can estimate these differences and perform tests.

```
. margins, at(fines=(8 9 10 11)) contrast(atcontrast(ar._at) nowald effects)
Contrasts of predictive margins                          Number of obs = 500
Model VCE: Robust

Expression: Mean function, predict()
1._at: fines =  8
2._at: fines =  9
3._at: fines = 10
4._at: fines = 11

Expression: Mean function, predict()
1._at: fines =  8
2._at: fines =  9
3._at: fines = 10
4._at: fines = 11

------------------------------------------------------------------------------
             |            Delta-method
             |   Contrast   std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         _at |
   (2 vs 1)  |   -21.2308   1.610261   -13.18   0.000    -24.38685   -18.07475
   (3 vs 2)  |   -7.94991   .7085254   -11.22   0.000    -9.338595   -6.561226
   (4 vs 3)  |  -5.620773   .5683614    -9.89   0.000    -6.734741   -4.506806
------------------------------------------------------------------------------
```

When fines are increased from $8,000 to $9,000, we expect a decrease of 21.23 in the number of citations. Smaller but still statistically significant decreases in the number of citations are expected as fines are increased from $9,000 to $10,000 and from $10,000 to $11,000.

◁

▷ Example 4: Estimating the effect for different levels of jurisdiction size

Now, we estimate the effect of increasing `fines` for different jurisdiction sizes.

```
. margins csize, dydx(fines)
Average marginal effects                                 Number of obs = 500
Model VCE: Robust

Expression: Mean function, predict()
dy/dx wrt:  fines

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
fines        |
       csize |
      Small  |  -5.992484   .4491224   -13.34   0.000    -6.872747    -5.11222
     Medium  |  -7.740284   .4366709   -17.73   0.000    -8.596144   -6.884425
      Large  |  -10.20492    .564166   -18.09   0.000    -11.31067   -9.099178
------------------------------------------------------------------------------
```

If all jurisdictions were small but other characteristics were as they are observed, then we expect that the marginal effect of `fines` would be −5.99. We see that the effect is more extreme as the size of the jurisdiction increases. If all jurisdictions were large, we expect that the average marginal effect of `fines` would be −10.20.

We can further explore the effects of fines for different jurisdiction sizes by estimating the expected number of citations with fines at specific levels.

```
. margins csize, at(fines=(8(1)11))
  (output omitted)
```

To visualize the effect, we type `marginsplot`.

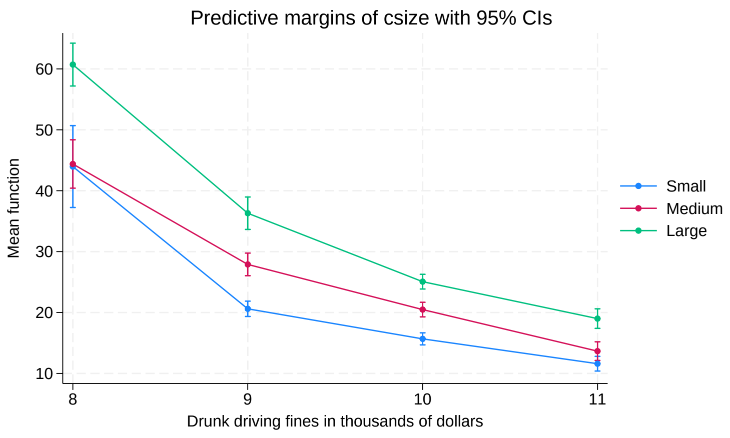

Figure 2.

For each jurisdiction size, we see that on average higher fines result in fewer citations. We also see that the effect of changing fine levels is nonlinear and differs across the counterfactual jurisdiction size. For instance, as fines increase from $8,000 to $9,000, the expected number of citations decreases faster for small jurisdictions than for medium ones.

◁

Stored results

`npregress series` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(r2)`	R^2
`e(r2_a)`	adjusted R^2
`e(converged)`	`1` if converged, `0` otherwise
`e(order)`	order of basis function
`e(rank)`	rank of `e(V)`

Macros

`e(cmd)`	`npregress`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(basis)`	`bsplines`, `splines`, or `polynomials`
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(title)`	title in estimation output
`e(vce)`	*vcetype* specified in `vce()`
`e(vcetype)`	title used to label Std. err.
`e(knots)`	number of knots selected
`e(datasignaturevars)`	variables used in calculation of checksum
`e(datasignature)`	the checksum
`e(estat_cmd)`	program used to implement `estat`
`e(predict)`	program used to implement `predict`
`e(properties)`	`b V`
`e(marginsok)`	predictions allowed by `margins`
`e(marginsprop)`	signals to the `margins` command
`e(marginsnotok)`	predictions disallowed by `margins`

Matrices	
`e(b)`	coefficient vector
`e(V)`	variance–covariance matrix of estimators
`e(V_modelbased)`	model-based variance
`e(ilog)`	iteration log (up to 20 iterations)
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:

Overview

The regression model of outcome y_i given the k-dimensional vector of covariates $\mathbf{x}_i$ was defined in (1) and (2) of *Remarks and examples* and is repeated here:

$$y_i = g\left(\mathbf{x}_i\right) + \varepsilon_i \tag{1}$$
$$E\left(\varepsilon_i|\mathbf{x}_i\right) = 0 \tag{2}$$

where ε_i is the error term. The covariates may include discrete and continuous variables. Equations (1) and (2) imply that

$$E\left(y_i|\mathbf{x}_i\right) = g\left(\mathbf{x}_i\right)$$

As discussed in *Remarks and examples*, series estimators have the form of ordinary least squares. Thus, we can write the estimate of the mean function as

$$\widehat{E}\left(y_i|\mathbf{x}_i\right) = \mathbf{z}\left(\mathbf{x}_i\right)\widehat{\boldsymbol{\beta}} \tag{5}$$

where $\mathbf{z}(\mathbf{x}_i)$ is a known q-dimensional vector for which every one of the q terms is a function of the k-dimensional vector of covariates $\mathbf{x}_i$. Let n be the sample size. If we define $\mathbf{Z}$ as the $n \times q$ matrix formed by the $\mathbf{z}(\mathbf{x}_i)$ for each individual i, then the q-dimensional coefficient vector $\widehat{\boldsymbol{\beta}}$ is the ordinary least-squares vector that comes from regressing the $n \times 1$ outcome vector $\mathbf{y}$ on $\mathbf{Z}$ and has the known form

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{Z}'\mathbf{Z}\right)^{-1}\left(\mathbf{Z}'\mathbf{y}\right) \tag{6}$$

Each one of the series estimators has a different form for $\mathbf{z}(\mathbf{x}_i)$. Below, we define $\mathbf{z}(\mathbf{x}_i)$ for polynomials, piecewise polynomial splines, and B-splines.

Polynomials

For a polynomial of order 1 with k continuous covariates, $\mathbf{x}_i \equiv (x_{i1}, x_{i2}, \ldots, x_{ik})$, $\mathbf{z}(\mathbf{x}_i)$ is

$$\mathbf{z}\left(\mathbf{x}_i\right) = \left(x_{i1}, x_{i2}, \ldots, x_{ik}\right) \tag{$P1$}$$

For notational convenience, we will refer to the polynomial above as $P1$, which also corresponds to the name we gave to the equation. More formally, we could have written $(x_{i1}, x_{i2}, \ldots, x_{ik}) \equiv P1$. We will maintain this notational convention below.

A polynomial of order 2 with k continuous covariates includes $P1$ and second-order terms:

$$\mathbf{z}\left(\mathbf{x}_i\right) = \left(P1, x_{i1}^2, x_{i1}x_{i2}, \ldots, x_{i1}x_{ik}, x_{i2}x_{i3}, \ldots, x_{ik}^2\right) \tag{$P2$}$$

A third-order polynomial with k continuous covariates includes the terms in $P2$ (which already includes $P1$) and third-order terms:

$$\mathbf{z}\left(\mathbf{x}_i\right) = \left(P2, x_{i1}^3, x_{i1}^2x_{i2}, \ldots, x_{i1}^2x_{ik}, x_{i1}x_{i2}x_{i3}, \ldots, x_{ik}^3\right) \tag{$P3$}$$

This recursive relationship continues. Thus, fourth-order polynomials includes the terms in $P3$ (which already includes $P1$ and $P2$) plus fourth-order terms.

For the polynomials above and all series estimators below, discrete covariates enter the model in levels, and each level is interacted with all other covariates in the model.

Piecewise polynomial splines

Piecewise polynomial splines are formed by a polynomial and functions of the form

$$\max(x_{ik} - t_{1k}, 0)$$

In the expression above, t_{1k} is a constant that is called a knot. The subscript of t_{1k} indicates that it is the first knot of the continuous covariate $\mathbf{x}_k$. The $\max(\cdot)$ function is 0 when $x_{ik} < t_{1k}$ and is $x_{ik} - t_{1k}$ otherwise. `npregress series` selects a set of knots for each one of the continuous covariates.

The regressors for `npregress series` using a piecewise polynomial spline of order 3 with one continuous covariate, $\mathbf{x}_1$, and k knots, $t_{11} < t_{21} < \cdots < t_{k1}$, are given by

$$\begin{aligned}\mathbf{z}\left(x_{i1}\right) = \Big\{ & x_{i1}, x_{i1}^2, x_{i1}^3, \max\left(x_{i1} - t_{11}, 0\right)^3, \max\left(x_{i1} - t_{21}, 0\right)^3, \ldots, \\ & \max\left(x_{i1} - t_{k1}, 0\right)^3 \Big\}\end{aligned} \tag{$S1$}$$

Equivalently, for another continuous covariate, $\mathbf{x}_2$, and k knots, $t_{12} < t_{22} < \cdots < t_{k2}$, we have

$$\begin{aligned}\mathbf{z}\left(x_{i2}\right) = \Big\{ & x_{i2}, x_{i2}^2, x_{i2}^3, \max\left(x_{i2} - t_{12}, 0\right)^3, \max\left(x_{i2} - t_{22}, 0\right)^3, \ldots, \\ & \max\left(x_{i2} - t_{k2}, 0\right)^3 \Big\}\end{aligned} \tag{$S2$}$$

To get $\mathbf{z}(x_{i1}, x_{i2})$ for $\mathbf{x}_1$ and $\mathbf{x}_2$ and k knots, we include all terms in $S1$ and $S2$ as well as all the terms that result from the interaction of their terms. We write it succinctly as

$$\mathbf{z}(x_{i1}, x_{i2}) = \{S1, S2, (S1)(S2)\} \tag{$S12$}$$

The description above refers to the default third-order piecewise polynomial spline. Below, we describe the cases for piecewise polynomial splines of order 1 and order 2. Going back to the one covariate case, if we want a piecewise polynomial spline of order 1 with k knots, we have

$$\mathbf{z}(x_{i1}) = \{x_{i1}, \max(x_{i1} - t_{11}, 0), \max(x_{i1} - t_{21}, 0), \ldots, \max(x_{i1} - t_{k1}, 0)\}$$

And for order 2 with k knots and one covariate, we have

$$\mathbf{z}(x_{i1}) = \left\{x_{i1}, x_{i1}^2, \max(x_{i1} - t_{11}, 0)^2, \max(x_{i1} - t_{21}, 0)^2, \ldots, \max(x_{i1} - t_{k1}, 0)^2\right\}$$

If we have more than one covariate, the logic of interacting the expressions for each covariate is the same as the logic we used for the third-order piecewise polynomial spline in $(S12)$.

To construct $\mathbf{z}(\mathbf{x}_i)$, continuous covariates are rescaled to be between 0 and 1 with the expression

$$\{\mathbf{x}_i - \min(\mathbf{x}_i)\}\left\{\frac{1}{\max(\mathbf{x}_i) - \min(\mathbf{x}_i)}\right\}$$

This rescaling is used to construct the piecewise polynomial spline and B-spline bases.

B-splines

To construct a B-spline basis, we need to define knots that are on the interior of the range of the covariates and knots that are at the upper and lower limits of the range or outside the range. The number of knots that are not in the interior differs depending on the order of the B-spline. For a B-spline of order 1 with k interior knots, $t_1, t_2, \ldots, t_k$, we need 4 additional knots. The set of knots for a first-order B-spline is therefore

$$t_{-1}, t_0, t_1, \ldots, t_k, t_{k+1}, t_{k+2}$$

We added t_{-1}, t_0, t_{k+1}, and t_{k+2} to the interior knots. By convention, $t_{-1} = t_0$ and $t_{k+1} = t_{k+2}$.

For a B-spline of order 2 with k interior knots, we need 6 additional knots. The set of knots is

$$t_{-2}, t_{-1}, t_0, t_1, \ldots, t_k, t_{k+1}, t_{k+2}, t_{k+3}$$

For a B-spline of order 3, the set of knots is

$$t_{-3}, t_{-2}, t_{-1}, t_0, t_1, \ldots, t_k, t_{k+1}, t_{k+2}, t_{k+3}, t_{k+4}$$

We first define a first-order B-spline for one continuous covariate $\mathbf{x}$ with k interior knots. Let $\mathbf{t}_j$ denote an $n \times 1$ vector for which all elements take the value of the jth knot t_j. The B-spline basis is formed by $k+2$ functions of the form

$$\begin{aligned} B_{j,1} = {} & \frac{(\mathbf{x} - t_j)}{t_{j+1} - t_j}\mathbf{1}\left(\mathbf{t}_j \leq \mathbf{x} < \mathbf{t}_{j+1}\right) + \frac{(t_{j+2} - \mathbf{x})}{t_{j+2} - t_{j+1}}\mathbf{1}\left(\mathbf{t}_{j+1} \leq \mathbf{x} < \mathbf{t}_{j+2}\right) \\ j = {} & -1, 0, 1, 2, \ldots, k \end{aligned} \tag{7}$$

Above, we use the indicator function $\mathbf{1}(\cdot)$, which is 1 when the condition inside the parentheses is satisfied and is 0 otherwise. Also, any term for which $t_{j+1} = t_j$ or $t_{j+2} = t_{j+1}$ is considered to be a vector of 0s.

The function $\mathbf{z}(\mathbf{x})$ used to estimate $g(\mathbf{x})$ is given by

$$\mathbf{z}(\mathbf{x}) = \left(B_{-1,1}, B_{0,1}, B_{1,1}, \ldots, B_{k,1}\right)$$

We now define a second-order B-spline for one continuous covariate $\mathbf{x}$ with k interior knots. The basis is constructed using the relationship given by

$$\begin{aligned} B_{j,2} = {} & \frac{(\mathbf{x} - t_j)}{t_{j+2} - t_j}B_{j,1} + \frac{(t_{j+3} - \mathbf{x})}{t_{j+3} - t_{j+1}}B_{j+1,1} \\ j = {} & -2, -1, 0, 1, 2, \ldots, k \end{aligned}$$

where $B_{j,1}$ and $B_{j+1,1}$ come from (7) above. Thus, second-order B-splines are a function of first-order B-splines, and as we will see below, third-order B-splines are a function of second-order B-splines. This recursion continues into higher orders, but `npregress series` stops at B-splines of order 3.

The function $\mathbf{z}(\mathbf{x})$ for the second-order B-spline is given by

$$\mathbf{z}(\mathbf{x}) = \left(B_{-2,2}, B_{-1,2}, B_{0,2}, B_{1,2}, \ldots, B_{k,2}\right)$$

The terms of the basis for a third-order B-spline are given by

$$\begin{aligned} B_{j,3} = {} & \frac{(\mathbf{x} - t_j)}{t_{j+3} - t_j}B_{j,2} + \frac{(t_{j+4} - \mathbf{x})}{t_{j+4} - t_{j+1}}B_{j+1,2} \\ j = {} & -3, -2, -1, 0, 1, 2, \ldots, k \end{aligned}$$

and the function $\mathbf{z}(\mathbf{x})$ for the third-order B-spline is

$$\mathbf{z}(\mathbf{x}) = \left(B_{-3,3}, B_{-2,3}, B_{-1,3}, B_{0,3}, B_{1,3}, \ldots, B_{k,3}\right) \tag{$B1$}$$

As was the case with piecewise polynomial splines, when there is more than one covariate, you include all functions of the form $(B1)$ and their interactions to form expressions like the one in $(S12)$.

Model selection

Below, we define the criteria used for model selection. In the case of B-splines and piecewise polynomial splines, `npregress series` selects the number of knots to be used for estimation. In the case of a polynomial basis, `npregress series` selects the order of the polynomial.

Let us first define the squared residuals, e_i^2, where $e_i = y_i - \widehat{g}\left(\mathbf{x}_i\right)$ and $\widehat{g}(\cdot)$ is the mean function estimate defined in (5). We denote the residuals for the regressions below as $e_i\left(\mathbf{t}_k\right)$ instead of e_i to signal that the estimates we obtain are a function of the set of knots, $\mathbf{t}_k$, used. In the case of polynomials, $\mathbf{t}_k$ will refer to the degree of the polynomial instead of knots.

Cross-validation

The cross-validation criterion, $\text{CV}\left(\mathbf{t}_k\right)$, is defined by

$$\text{CV}\left(\mathbf{t}_k\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{e_i\left(\mathbf{t}_k\right)^2}{\left(1-h_{ii}\right)^2} \tag{8}$$

In (8), h_{ii} are the diagonal elements of the matrix $\mathbf{Z}\left(\mathbf{Z}'\mathbf{Z}\right)\mathbf{Z}'$, where Z is defined in (6) above and n is the size of the estimation sample.

`npregress series` computes $\text{CV}\left(\mathbf{t}_k\right)$ for different sets of knots, $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k, \ldots,$ where $\mathbf{t}_1 \subset \mathbf{t}_2 \subset \ldots \subset \mathbf{t}_k \subset \ldots$, and then selects the model with the smallest value for the cross-validation criterion.

Generalized cross-validation

The generalized cross-validation criterion, $\text{GCV}\left(\mathbf{t}_k\right)$, is given by

$$\text{GCV}\left(\mathbf{t}_k\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{e_i\left(\mathbf{t}_k\right)^2}{\left\{1-(K/n)\right\}^2}$$

where K is the number of estimated parameters and the other arguments are equivalent to those defined in (8). As with cross-validation, $\text{GCV}\left(\mathbf{t}_k\right)$ is computed for a set of models with an increasing number of nested knots, in the case of piecewise polynomial splines and B-splines, and of polynomial order in the case of polynomials. The minimum of the sequence is the selected model.

Mallows's C_p

$$\text{Mallows}\left(\mathbf{t}_k\right) = \frac{1}{n}\sum_{i=1}^{n} e_i\left(\mathbf{t}_k\right)^2 \left(1 + \frac{2K}{n}\right)$$

As with cross-validation, $\text{Mallows}\left(\mathbf{t}_k\right)$ is computed for a set of models with an increasing number of nested knots, and the minimum of the sequence is the selected model.

AIC and BIC

See *Methods and formulas* in [R] **estat ic**.

References

Chen, X. 2007. Large sample sieve estimation of semi-nonparametric models. In Vol. 6B of *Handbook of Econometrics*, ed. J. J. Heckman and E. E. Leamer, 5549–5632. Amsterdam: Elsevier. https://doi.org/10.1016/S1573-4412(07)06076-X.

Chetverikov, D., D. Kim, and D. Wilhelm. 2018. Nonparametric instrumental-variable estimation. *Stata Journal* 18: 937–950.

de Boor, C. 2001. *A Practical Guide to Splines*. Rev. ed. New York: Springer.

Eubank, R. L. 1999. *Nonparametric Regression and Spline Smoothing*. 2nd ed. New York: Dekker.

Hansen, B. E. 2009. University of Wisconsin–Madison, ECON 718, NonParametric Econometrics, Spring 2009, course notes. Last visited on 2019/01/15. https://www.ssc.wisc.edu/~bhansen/718/718.htm.

——. 2018. Econometrics. https://www.ssc.wisc.edu/~bhansen/econometrics/Econometrics.pdf.

Li, J., Z. Liao, and M. Gao. 2020. Uniform nonparametric inference for time series using Stata. *Stata Journal* 20: 706–720.

Li, J., Z. Liao, R. Quaedvlieg, and W. Zhou. 2022. Conditional evaluation of predictive models: The cspa command. *Stata Journal* 22: 924–940.

Li, Q., and J. S. Racine. 2007. *Nonparametric Econometrics: Theory and Practice*. Princeton, NJ: Princeton University Press.

Newey, W. K. 1997. Convergence rates and asymptotic normality for series estimators. *Journal of Econometrics* 79: 147–168. https://doi.org/10.1016/S0304-4076(97)00011-0.

Schoenberg, I. J., ed. 1969. *Approximations with Special Emphasis on Spline Functions*. New York: Academic Press.

Schumaker, L. L. 2007. *Spline Functions: Basic Theory*. 3rd ed. Cambridge: Cambridge University Press.

Shao, J. 2003. *Mathematical Statistics*. 2nd ed. New York: Springer.

Also see

[R] **npregress series postestimation** — Postestimation tools for npregress series

[R] **npregress intro** — Introduction to nonparametric regression

[R] **kdensity** — Univariate kernel density estimation

[R] **lpoly** — Kernel-weighted local polynomial smoothing

[R] **makespline** — Spline generation

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