Nonparametric Methods in Stata

Eduardo García Echeverri

Stata Webinar, 2023

Outline

What are a nonparametric methods?

Estimation method that is agnostic about:

- **Probability distributions** of outcomes and covariates.
- **Functional forms** relating outcomes and covariates.

Unlike **parametric methods**, which require us to specify these two.

Example

What is the effect of **smoking during pregnancy** (`msmoke`) on the **babies weight** at birth (`bweight`)?

- **Parametric:**

$$\texttt{msmoke} = \beta_0 + \beta_1 \texttt{bweight} + \gamma \text{Controls} + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

- **Nonparametric:**

$$\texttt{msmoke} = g(\texttt{bweight}, \text{Controls}) + \varepsilon$$

$$\mathbb{E}\left[\varepsilon | \texttt{bweight}, \text{Controls}\right] = 0$$

The Appeal of Nonparametric Methods

Advantages:

1. Avoid the problems that arise with **misspecification**.
2. Improve predictions.
3. Knowing the functional form is not needed to answer important research questions.
4. Easily implementable in Stata.

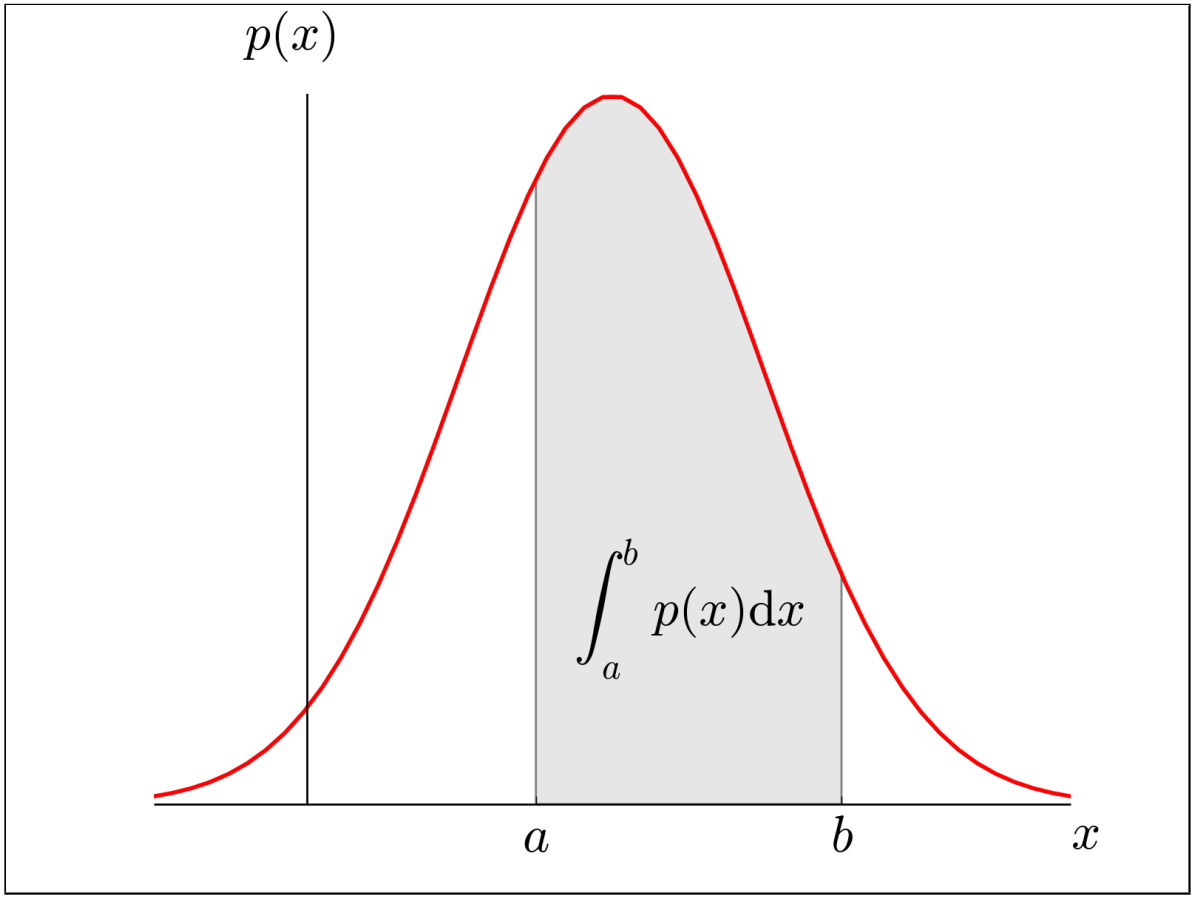
Disadvantages:

1. Data intensive (esp. with many covariates)
2. Computationally costly.

Outline

What is a probability density?

Probability densities tell us how a **random variable is distributed**.



Source: deepai.org

Why are they useful?

Probability densities can show us:

1. Which values of the variable are **likely/unlikely**.
2. The **mode(s)** of the distribution (**peaks** of the density)
3. The **range** of values of the variable
4. The probability of **extreme events** (tails of the distribution)

Parametric estimation of densities

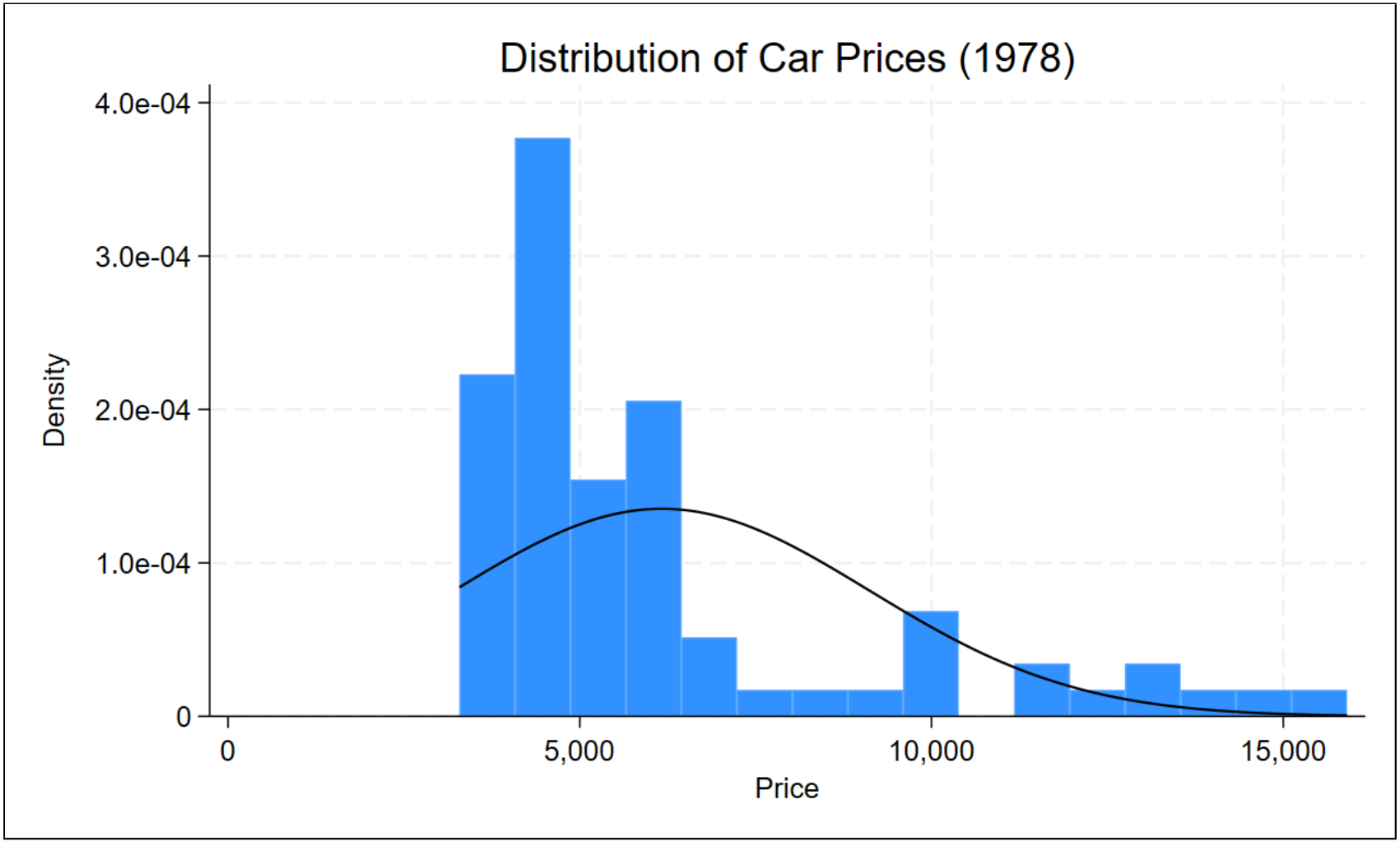
First, specify a **type** of probability density function:

1. **Normal** (most common choice)
2. Log-normal
3. ...

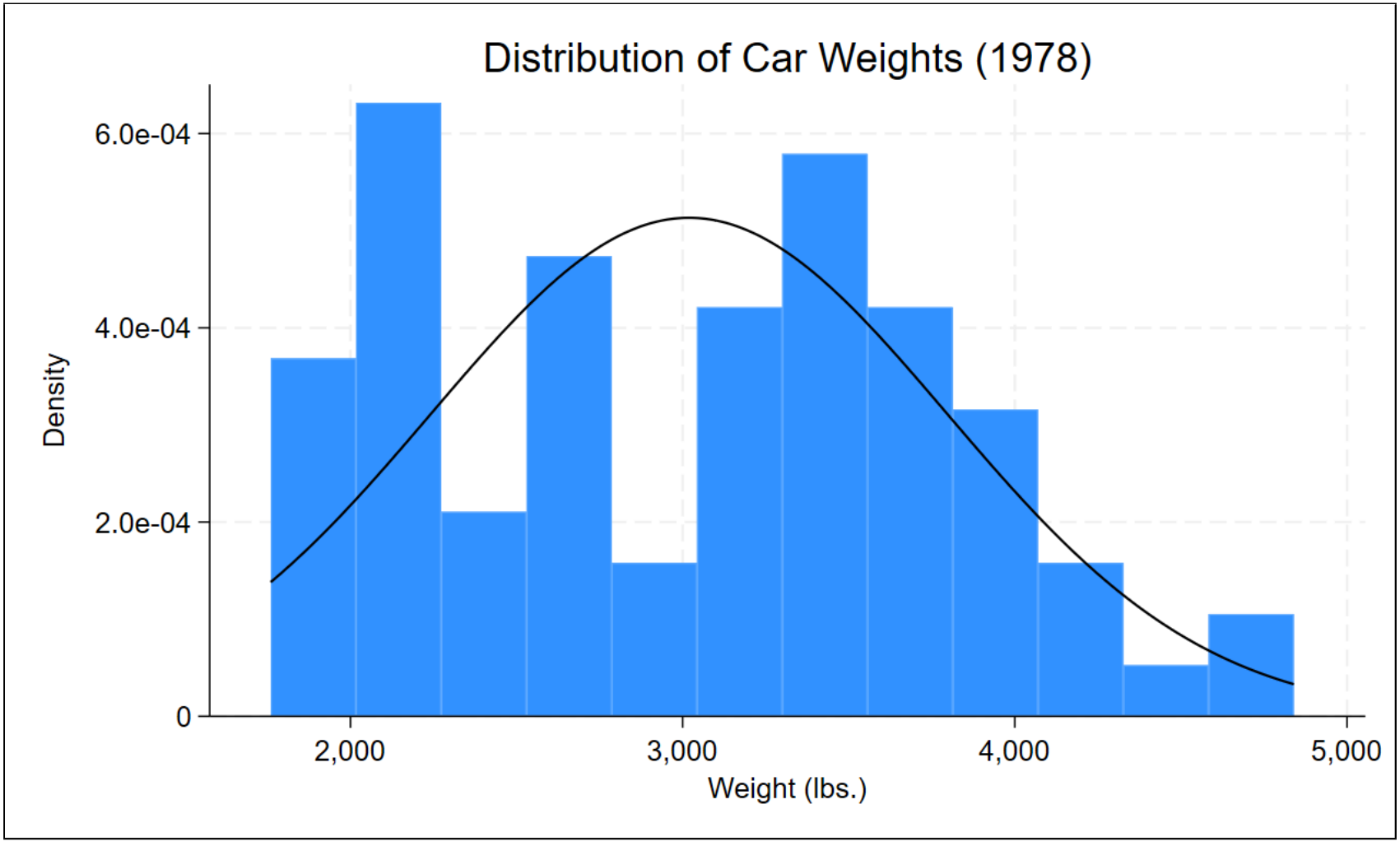
Second, based on **sample mean and variance**, **estimate** the density within the type that **best matches data**.

Problem: misspecification. Normal might be poor approximation.

Normality may not hold – fat right tails



Normality may not hold – multiple modes



Kernel Density Estimators

Kernel density estimators **don't assume a functional form**.

To estimate a **density** at point x:

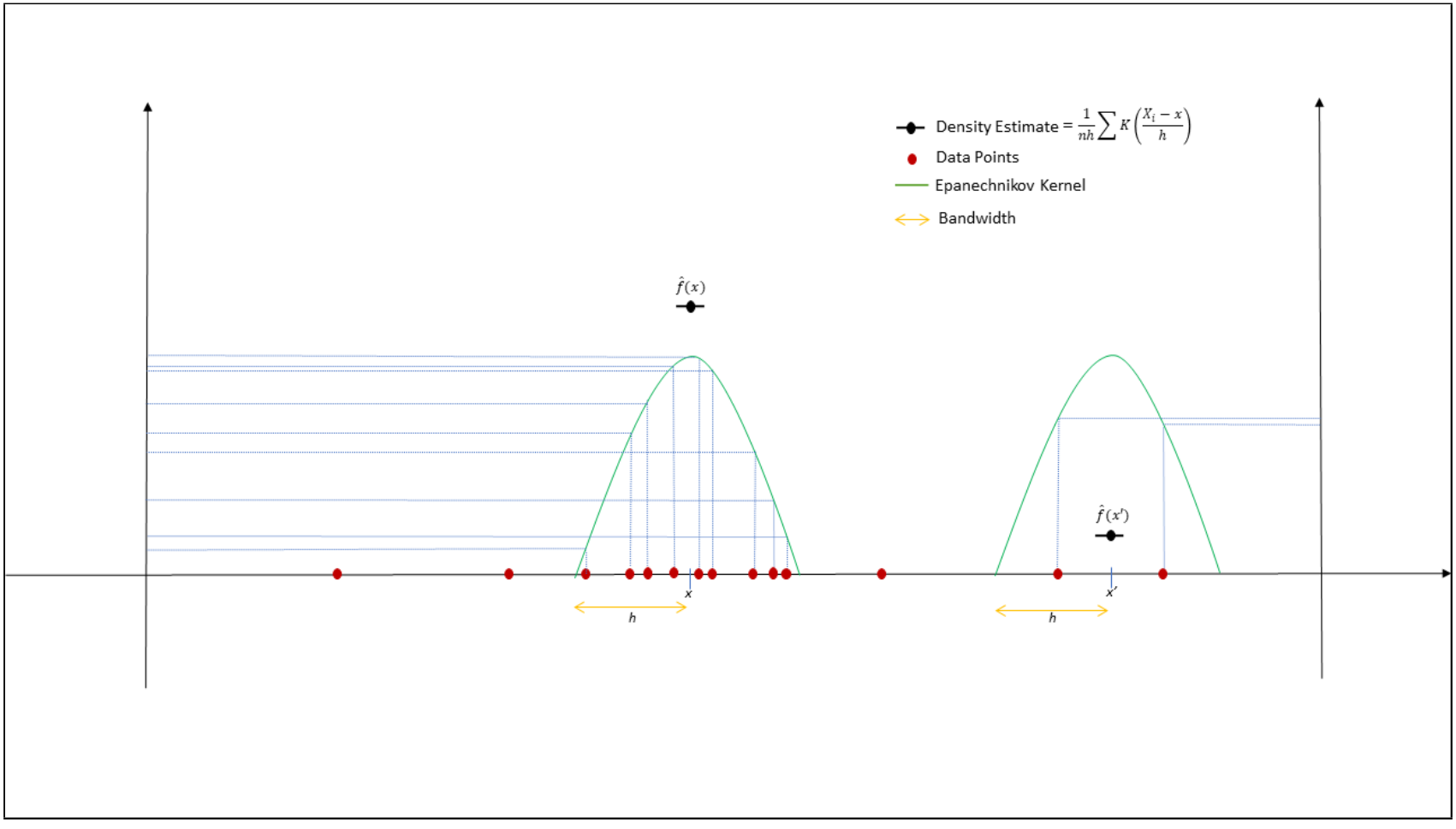
- **"Weighted count"** of how many data points are close to x,

$$\widehat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

h: **Bandwidth** – a positive number (very important!)

$K(\cdot)$: **Kernel** – a function ($K : \mathbb{R} \to \mathbb{R}$)

Illustration



Implementation using Stata

`kdensity` *varname [if] [in] [weight] [, options]*

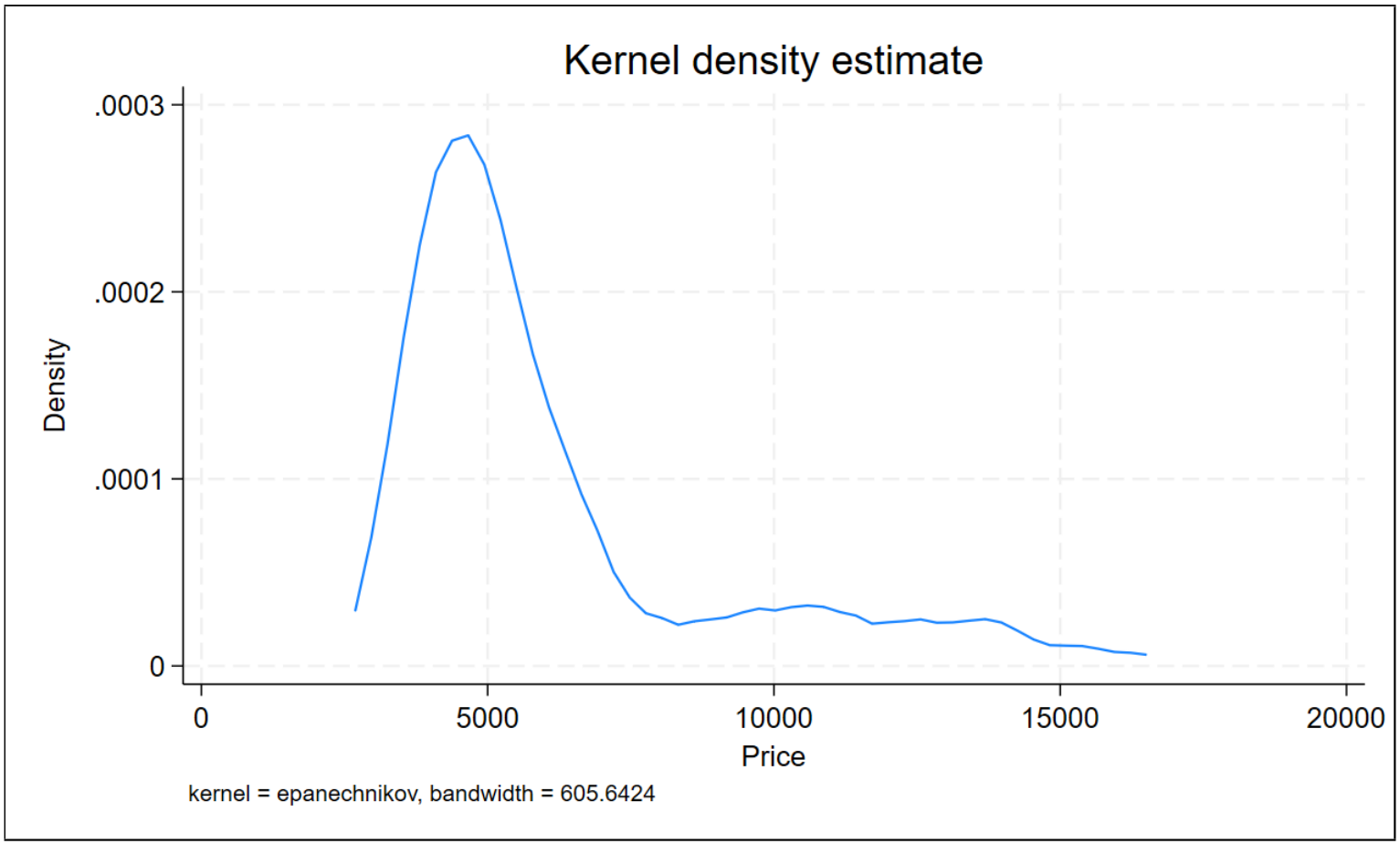
Quick Start:

Graph of the **kernel density estimate** for variable $\texttt{x}_1$

- `kdensity` $\texttt{x}_1$

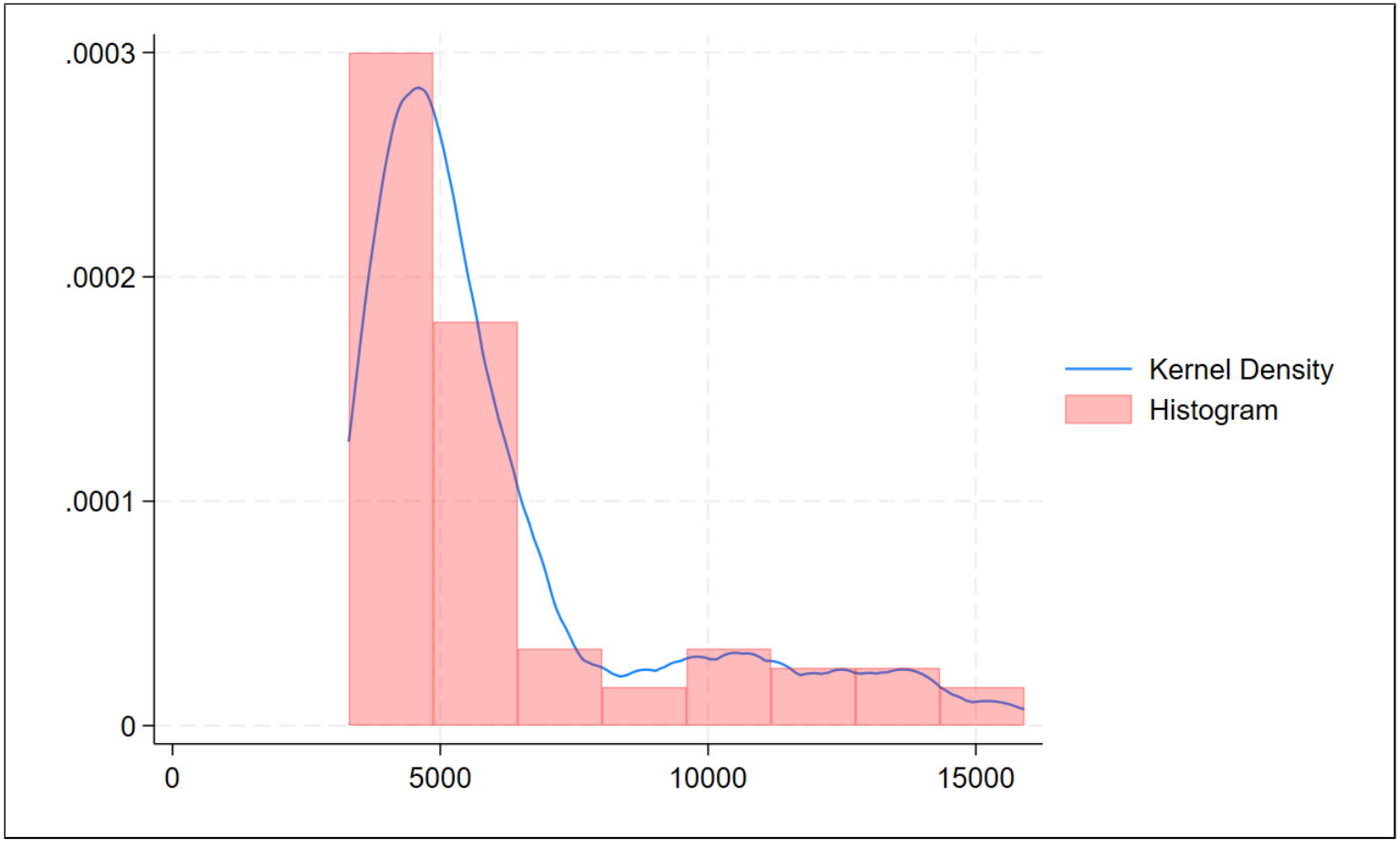
Example 1: Graphing a probability density

- `sysuse auto, clear`
- `kdensity price`



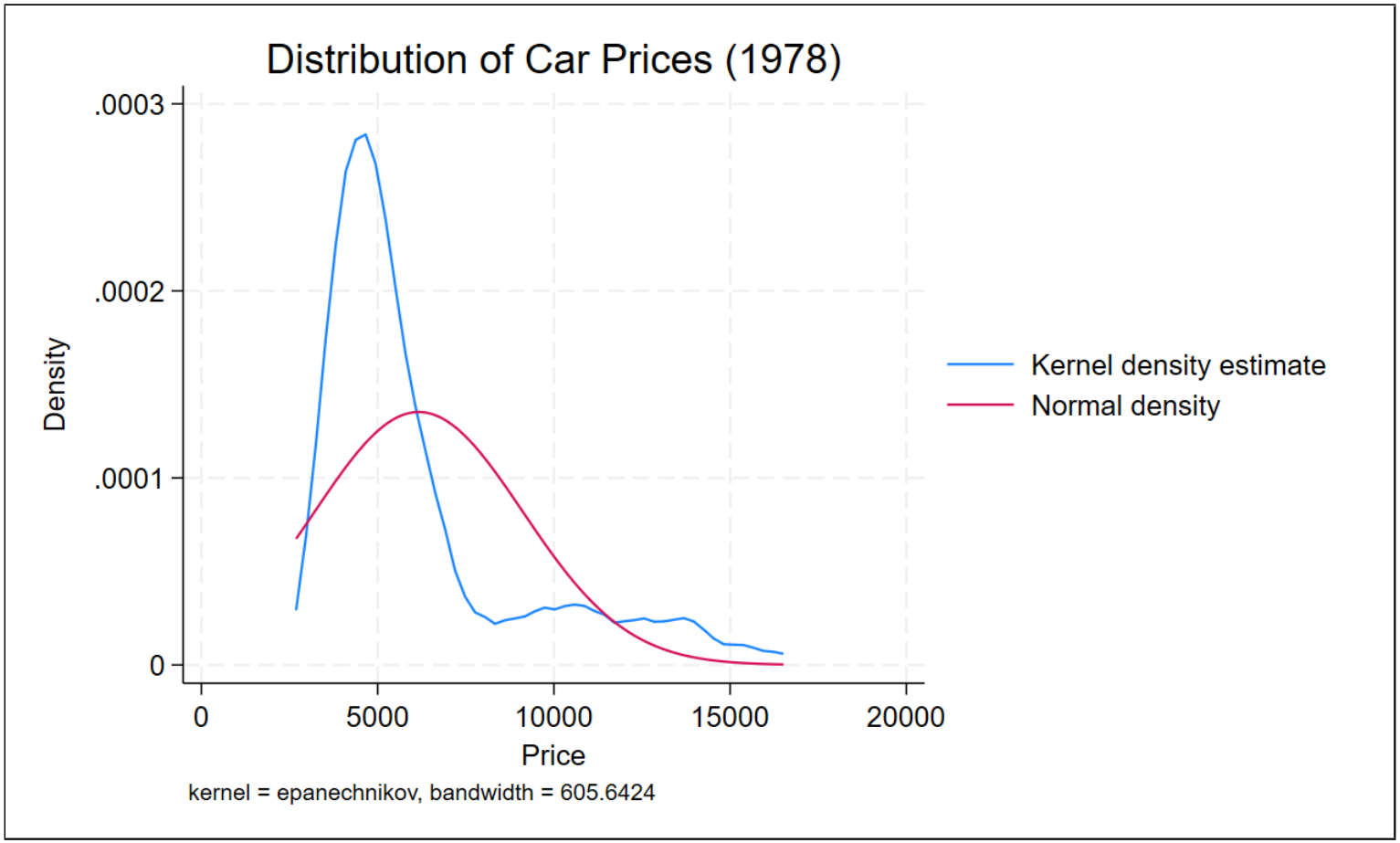
Example 1: Better fit the data

- `twoway kdensity price || histogram price, legend(order(1 "Kernel Density" 2 "Histogram")) color(red%30)`



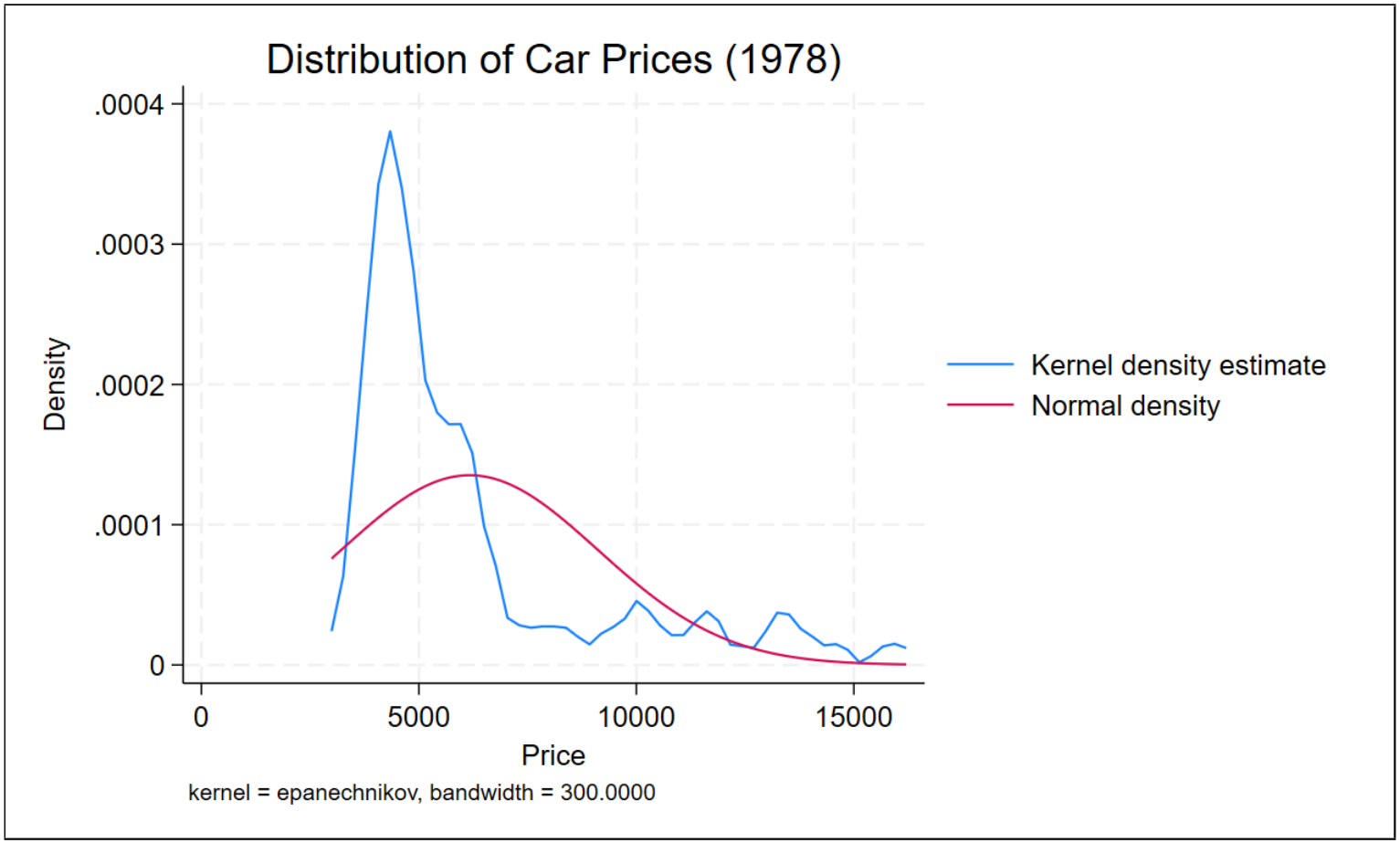
Example 2: Comparing to a normal distribution

- `kdensity price, normal title("Distribution of Car Prices")`



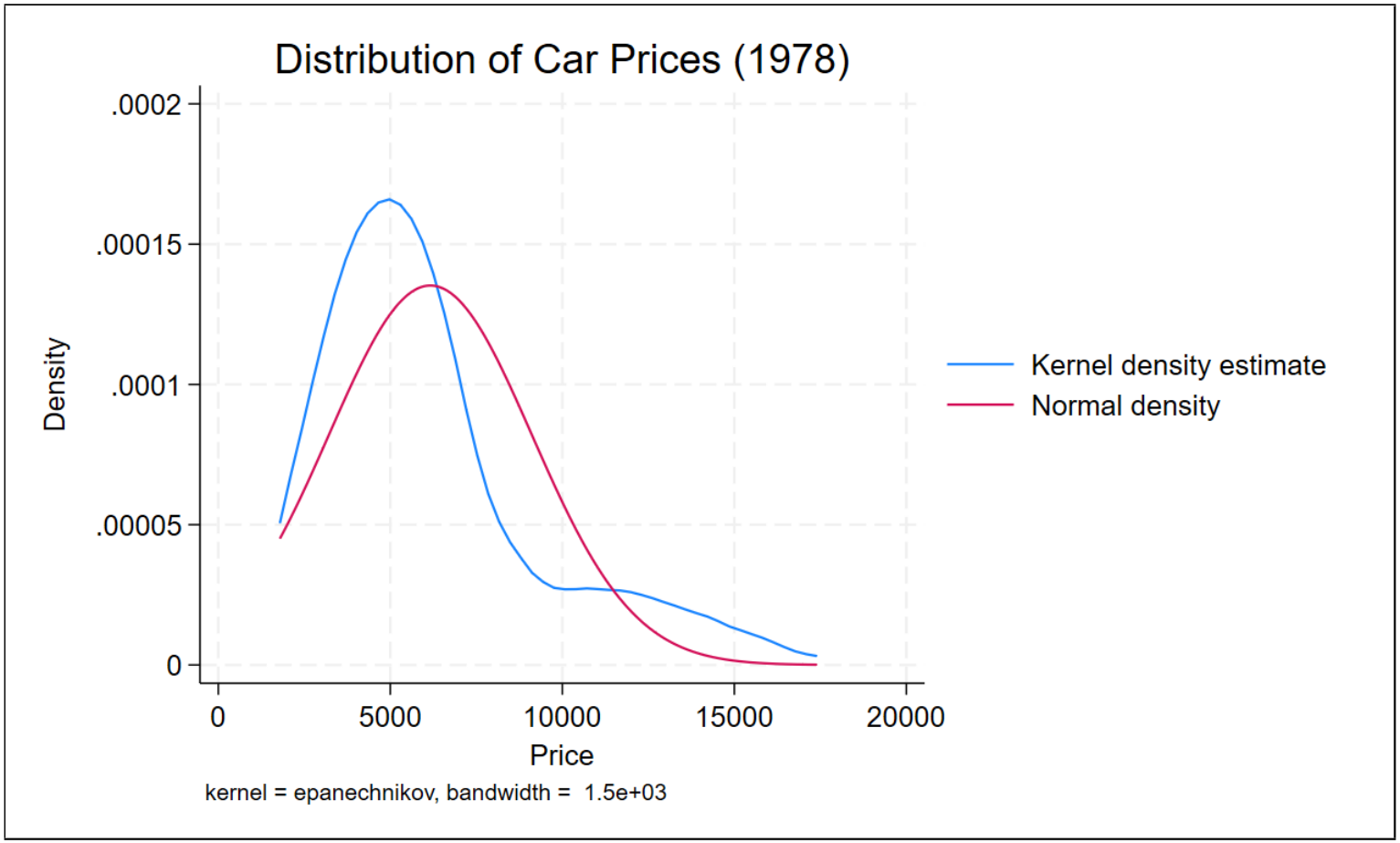
Example 3: Using a smaller bandwidth (overfitting)

- `kdensity price, normal bwidth(300) title("Distribution of Car Weights")`



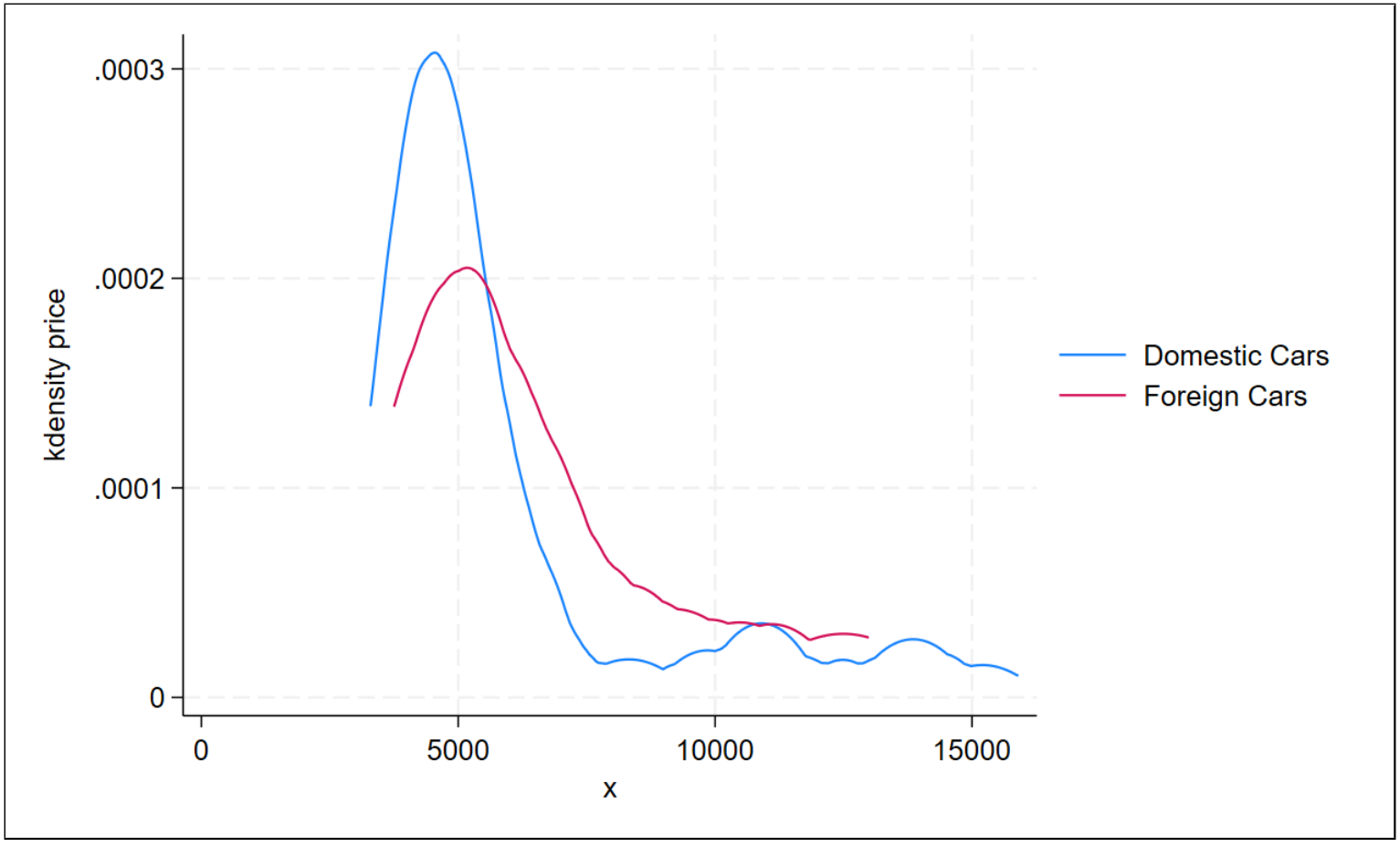
Example 4: Using a larger bandwidth (oversmoothing)

- `kdensity price, normal bwidth(1500) title("Distribution of Car Prices")`



Example 5: Comparing Price Distributions

- `twoway kdensity price if foreign==0 || kdensity price if foreign==1`



Useful Options

`kernel:` specify the kernel function:

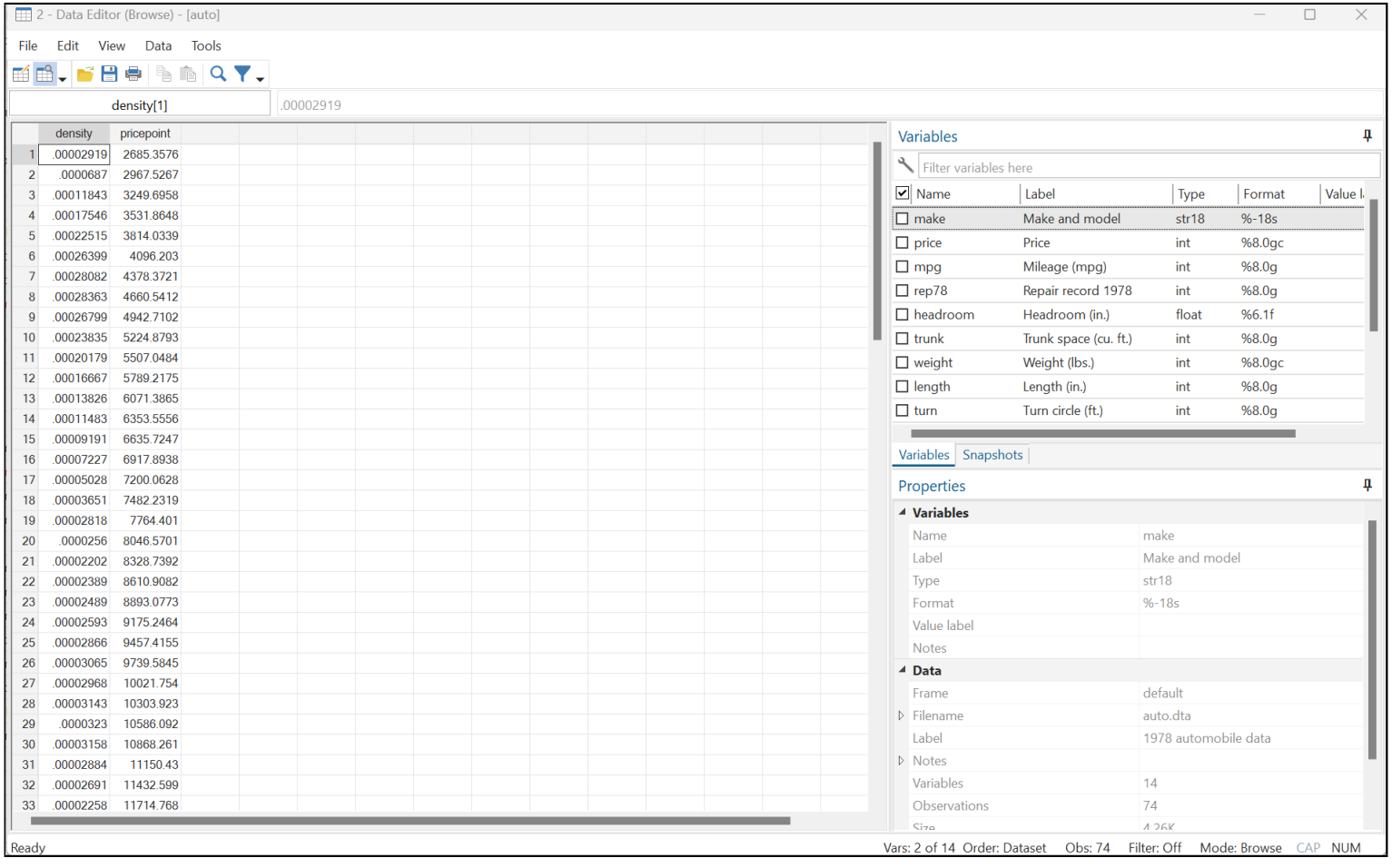
- epanechnikov; default
- gaussian
- triangle...

`generate(`*newvar*$_1$`,` *newvar*$_2$`):` store estimation points in *newvar*$_1$ and density estimates in *newvar*$_2$

`nograph:` Suppress the graph.

Option generate

- `kdensity price, generate(pricepoint density) nograph`
- `browse pricepoint density`



Outline

Conditional Expectations–Regression Function

Objective:
Estimate the **conditional mean/regression function** $g(\cdot)$:

$$y = g(X) + \varepsilon$$

$$\mathbb{E}[\varepsilon|X] = 0$$

Parametric approaches:

- Linear regression: $g(X) = x\beta$
- Probit: $g(X) = \Phi(x\beta)$
- Poisson: $g(X) = \exp(x\beta)$

Problems may arise due to **misspecification**.
Kernel regression doesn't assume a functional form for $g(\cdot)$

Outline

Kernel Regression – Local Linear

To estimate the **conditional expectation** at point x:

- **"Weighted regression"** of y on X using points close to x,

$$\min_{\gamma_0,\gamma_1} \sum_{i=1}^{n} \Big(y_i - \gamma_0 - \gamma_1(x_i - x)\Big)^2 \cdot K\Big(\frac{x - X_i}{h}\Big)$$

γ_0 : Predicted **conditional expectation** $\hat{g}(x)$

γ_1: Predicted **derivative** of $g(\cdot)$ at point x.

h: **Bandwidth** – a positive number (very important!)

$K(\cdot)$: **Kernel** – a function ($K : \mathbb{R} \to \mathbb{R}$)

Kernel Regression – Local Constant

To estimate the **conditional expectation** at point x:

- **"Weighted regression"** of y on X using points close to x,

$$\min_{\gamma_0,\gamma_1} \sum_{i=1}^{n} \Big(y_i - \gamma_0\Big)^2 \cdot K\Big(\frac{x - X_i}{h}\Big)$$

γ_0 : Predicted **conditional expectation** $\hat{g}(x)$

h: **Bandwidth** – a positive number (very important!)

$K(\cdot)$: **Kernel** – a function ($K : \mathbb{R} \to \mathbb{R}$)

Implementation in Stata

`npregress kernel` *depvar indepvars [if] [in] [, options]*

Quick Start:

Local-linear kernel regression of `y` on `x` and discrete covariate `a`:

- `npregress kernel y x i.a`

Local-const. kernel regression of `y` on `x` and discrete covariate `a`:

- `npregress kernel y x i.a, estimator(constant) noderivatives`

Example 7: The effect of smoking on babies' weight

Outcome:

- `bweight`: **Baby's weight** in grams.

Treatment:

- `msmoke`: **Cigarettes** smoked during pregnancy (4 categories)

Controls:

- `mage`: Mother's age.
- `medu`: Mother's educational attainment.
- `alcohol`: 1 if alcohol was consumed during pregnancy.
- `prenatal`: trimester of first prenatal visit.

Example 7: Local-Linear Kernel Regression

```
. npregress kernel bweight mage medu i.msmoke i.alcohol i.prenatal, nolog

Bandwidth
```

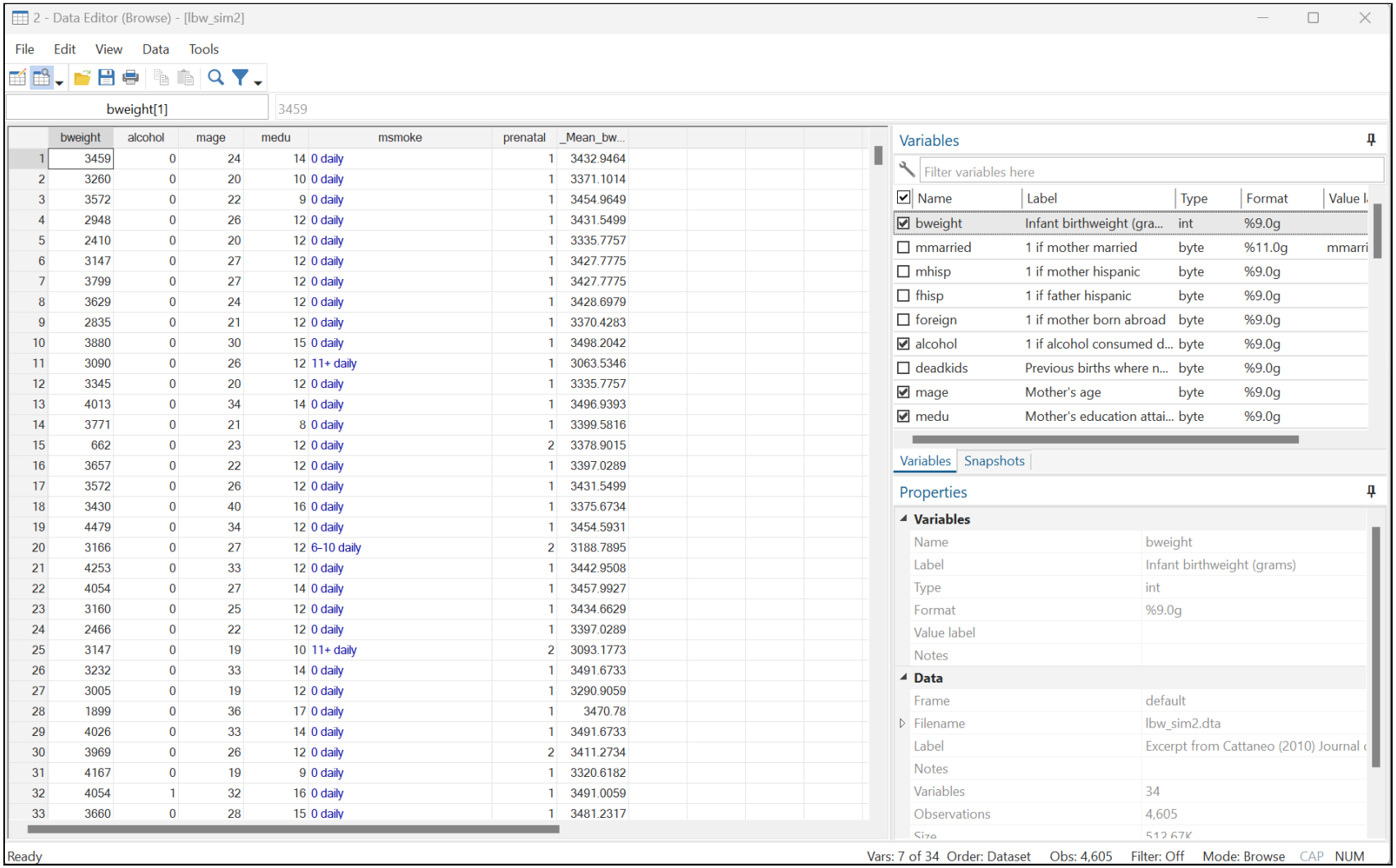
	Mean	Effect
mage	2.178108	3.09538
medu	.9284651	1.319472
msmoke	.5	.5
alcohol	.5	.5
prenatal	.5	.5

```
Local-linear regression                 Number of obs      =         4,605
Continuous kernel : epanechnikov        E(Kernel obs)      =         4,605
Discrete kernel   : liracine            R-squared          =        0.0893
Bandwidth         : cross-validation
```

bweight	Estimate
Mean	
bweight	3357.97
Effect	
mage	9.550276
medu	4.95285
msmoke	
(1-5 daily vs 0 daily)	-114.3037
(6-10 daily vs 0 daily)	-214.6567
(11+ daily vs 0 daily)	-306.8657
alcohol	
(1 vs 0)	-45.48752
prenatal	
(1 vs 0)	31.2373
(2 vs 0)	10.41102
(3 vs 0)	-30.41237

```
Note: Effect estimates are averages of derivatives for continuous covariates and averages of contrasts for factor covariates.
Note: You may compute standard errors using vce(bootstrap) or reps().
```

Example 7: Generated Predictions



Example 8: Bootstrap Standard Errors

```
. npregress kernel bweight mage medu i.msmoke i.alcohol i.prenatal, nolog reps(40) seed(1234)
(running npregress on estimation sample)

Bootstrap replications (40): .........10.........20.........30.........40 done

Bandwidth
```

	Mean	Effect
mage	2.178108	3.09538
medu	.9284651	1.319472
msmoke	.5	.5
alcohol	.5	.5
prenatal	.5	.5

```
Local-linear regression                 Number of obs      =         4,605
Continuous kernel : epanechnikov        E(Kernel obs)      =         4,605
Discrete kernel   : liracine            R-squared          =        0.0893
Bandwidth         : cross-validation
```

	Observed	Bootstrap			Percentile	
bweight	estimate	std. err.	z	P>|z|	[95% conf.	interval]
Mean						
bweight	3357.97	10.23615	328.05	0.000	3341.884	3379.431
Effect						
mage	9.550276	1.77653	5.38	0.000	6.437531	12.45403
medu	4.95285	7.087289	0.70	0.485	-9.00919	15.87284
msmoke						
(1-5 daily vs 0 daily)	-114.3037	10.67519	-10.71	0.000	-135.9948	-95.59864
(6-10 daily vs 0 daily)	-214.6567	19.9158	-10.78	0.000	-257.2621	-179.2783
(11+ daily vs 0 daily)	-306.8657	29.30562	-10.47	0.000	-367.6013	-261.8348
alcohol						
(1 vs 0)	-45.48752	42.34277	-1.07	0.283	-119.0151	39.6552
prenatal						
(1 vs 0)	31.2373	23.23159	1.34	0.179	-16.08415	73.16318
(2 vs 0)	10.41102	41.17059	0.25	0.800	-79.54673	84.2395
(3 vs 0)	-30.41237	61.28031	-0.50	0.620	-158.3454	86.92535

Note: Effect estimates are averages of derivatives for continuous covariates and averages of contrasts for factor covariates.

Example 9: Displaying Results Graphically

```
. npregress kernel bweight mage, nolog

Bandwidth
```

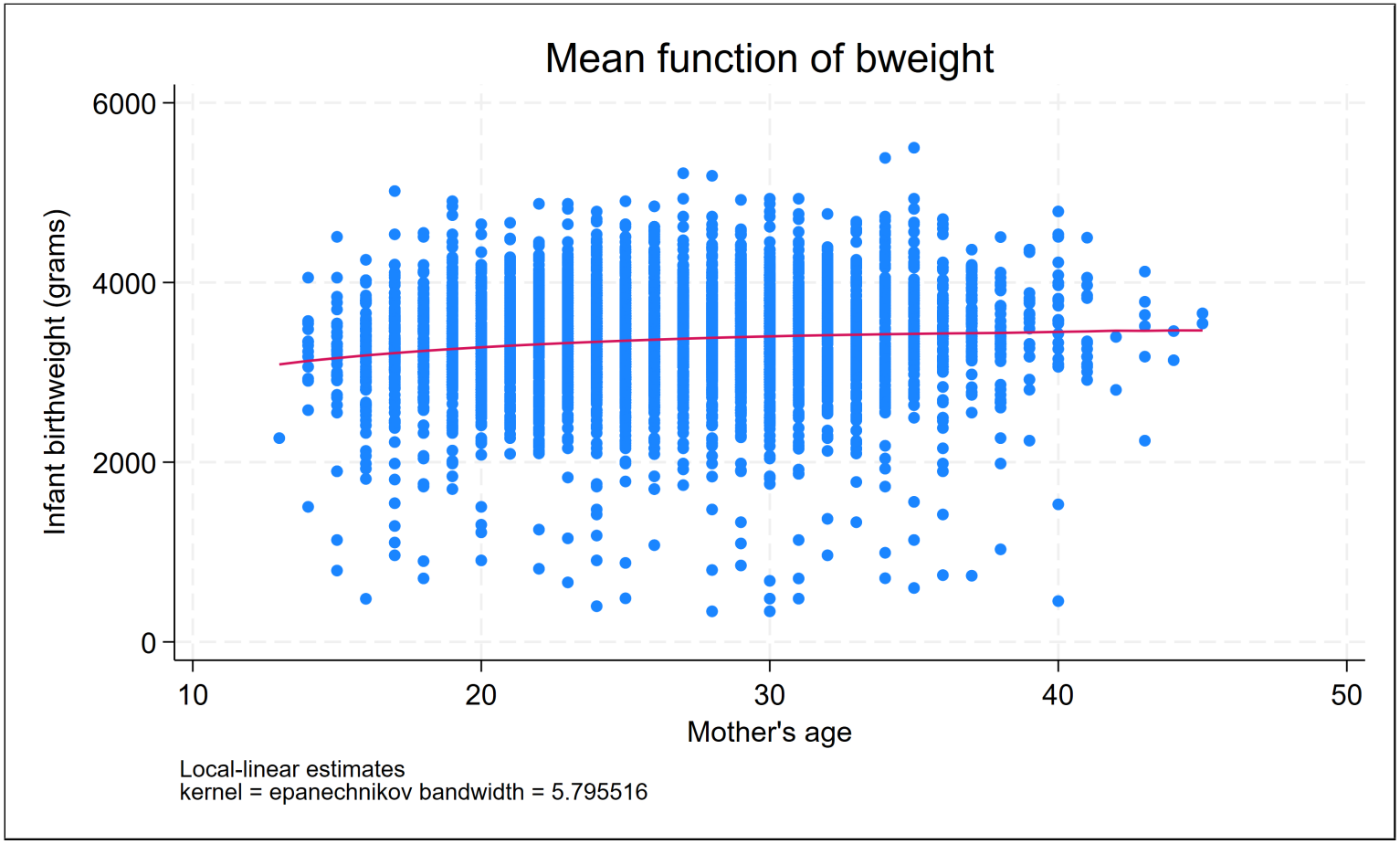
	Mean	Effect
mage	5.795516	3.695218

```
Local-linear regression                 Number of obs      =         4,605
Kernel    : epanechnikov                E(Kernel obs)      =         4,605
Bandwidth: cross-validation             R-squared          =        0.0132
```

bweight	Estimate
Mean	
bweight	3354.357
Effect	
mage	11.41642

```
Note: Effect estimates are averages of derivatives.
Note: You may compute standard errors using vce(bootstrap) or reps().

.
. npgraph
```

Example 9: Displaying Results Graphically



Outline

Series Approximations – Introduction

We can also **approximate function** $g(\cdot)$ **using series**. For instance, recall the Taylor expansion:

$$g(x) = g(0) + \frac{g'(0)}{1!} \cdot x + \frac{g''(0)}{2!} \cdot x^2 + \frac{g^{(3)}(0)}{3!} \cdot x^3 + \text{Remainder}$$

Thus, we can **estimate function** $g(\cdot)$ as:

$$\hat{g}(x_i) = z(x_i)\hat{\beta}$$

Where $z(x_i) = \left(1, x_i, x_i^2, x_i^3\right)$ and $\hat{\beta} = (Z^\intercal Z)^{-1} Z^\intercal y$.

Basis Supported in Stata

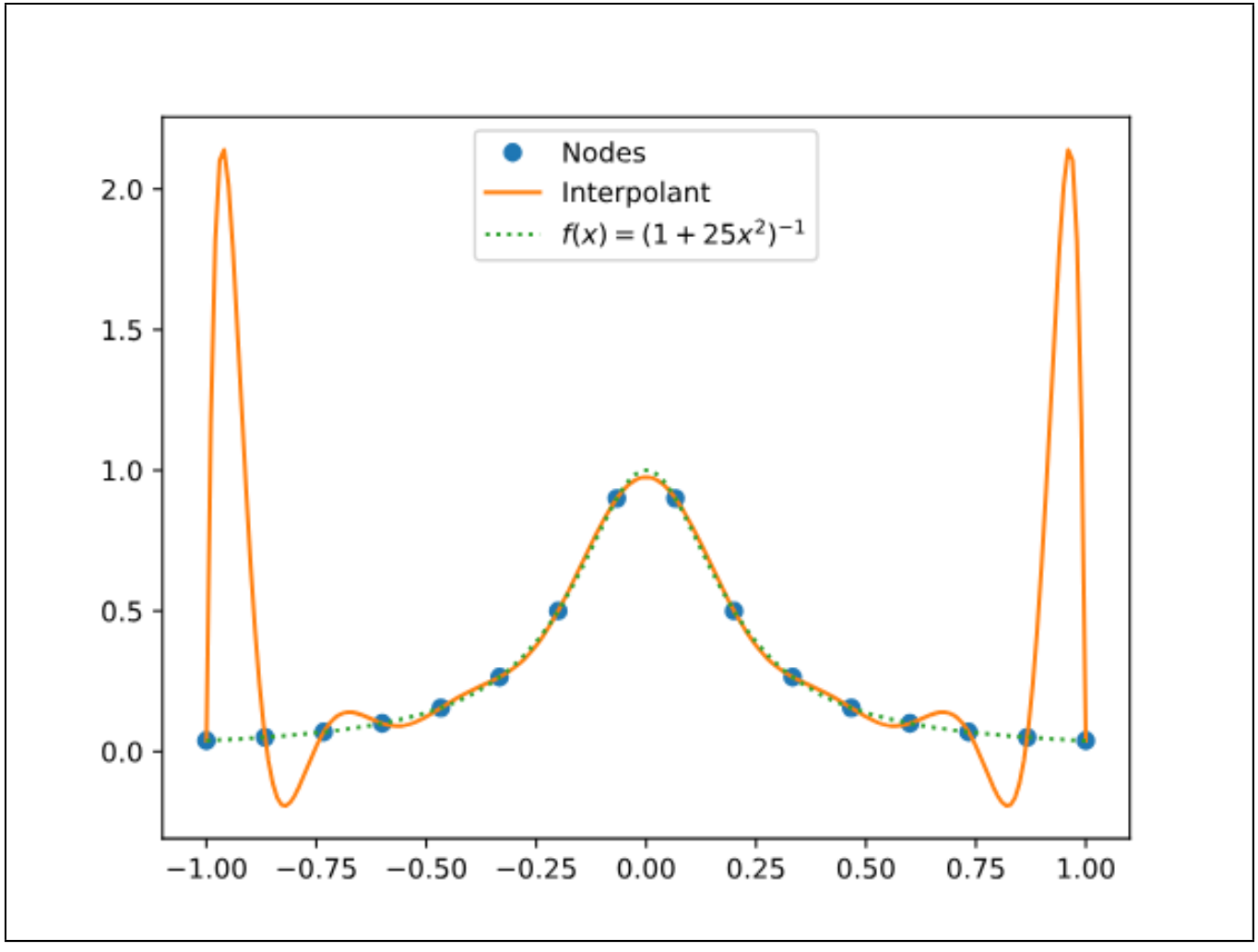
Polynomial Basis:
Function $g(\cdot)$ is approximated with a polynomial.

Piecewise Polynomial Spline Basis:
Function $g(\cdot)$ is approximated with a piecewise polynomial.

B-Spline Basis (default):
Function $g(\cdot)$ is approximated with a spline function. A spline is a smoothed piecewise polynomial.

Piecewise polynomials and splines **alleviate Runge's phenomenon**.

Runge's Phenomenon



Source: John D. Cook Consulting

Implementation in Stata

`npregress series` *depvar indepvars [if] [in] [weight] [, options]*

Quick Start:

Nonparametric regression of `y` on `x` and discrete covariate `a` (using B-spline basis):

- `npregress series y x i.a`

As above, but use a polynomial basis instead:

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

Example 10: Effect of fines on the number of DUI citations

Outcome:

- `citations`: **Annual DUI citations** in a county.

Treatment:

- `fines`: **Fines** for drunk driving in the county

Controls:

- `csize`: Size of the county (3 categories)
- `college`: 1 if there is a college in the county.

Nonparametric regression in Stata using series

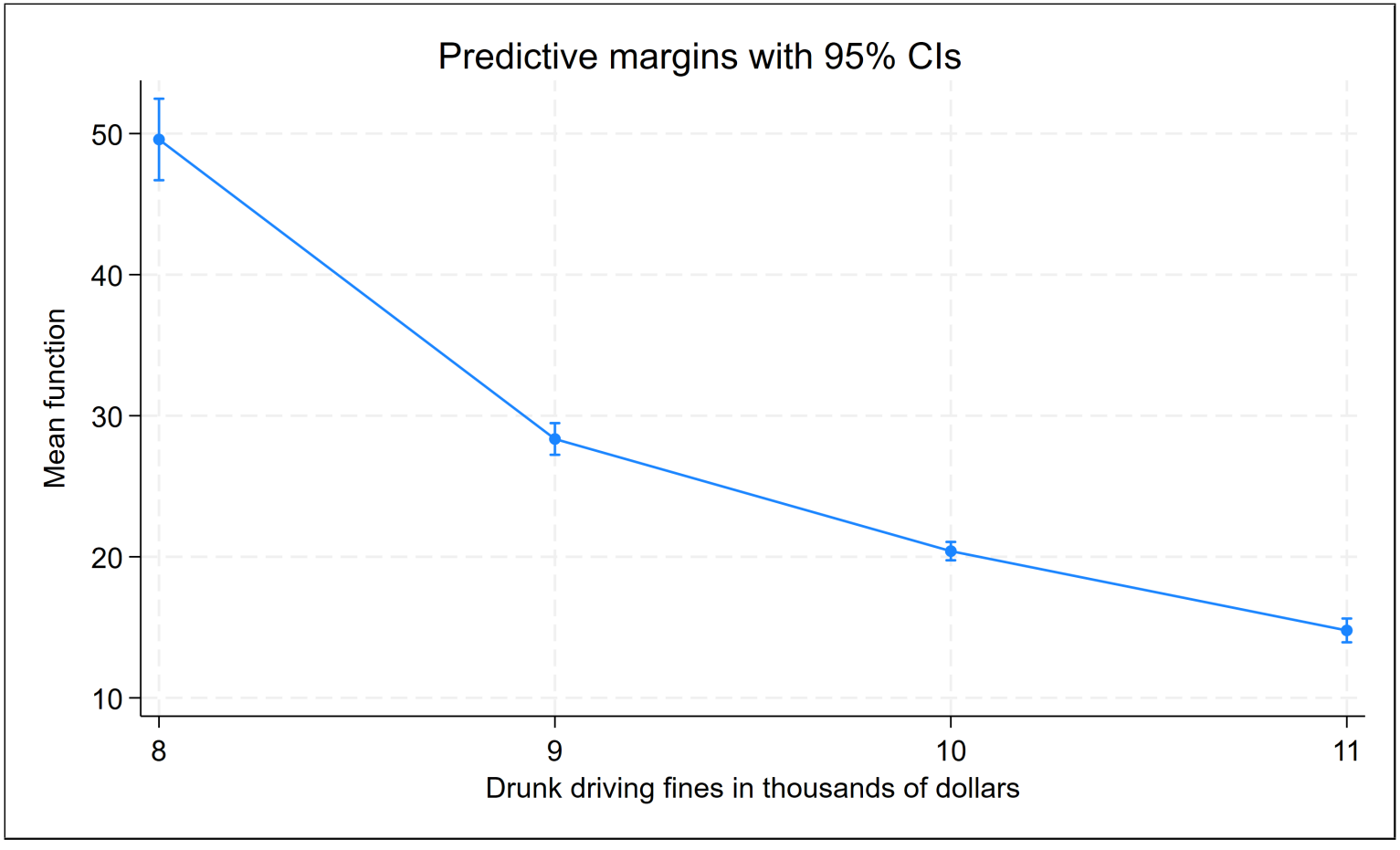
```
. npregress series citations fines i.csize i.college

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
```

citations	Effect	Robust 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 college)	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.
```

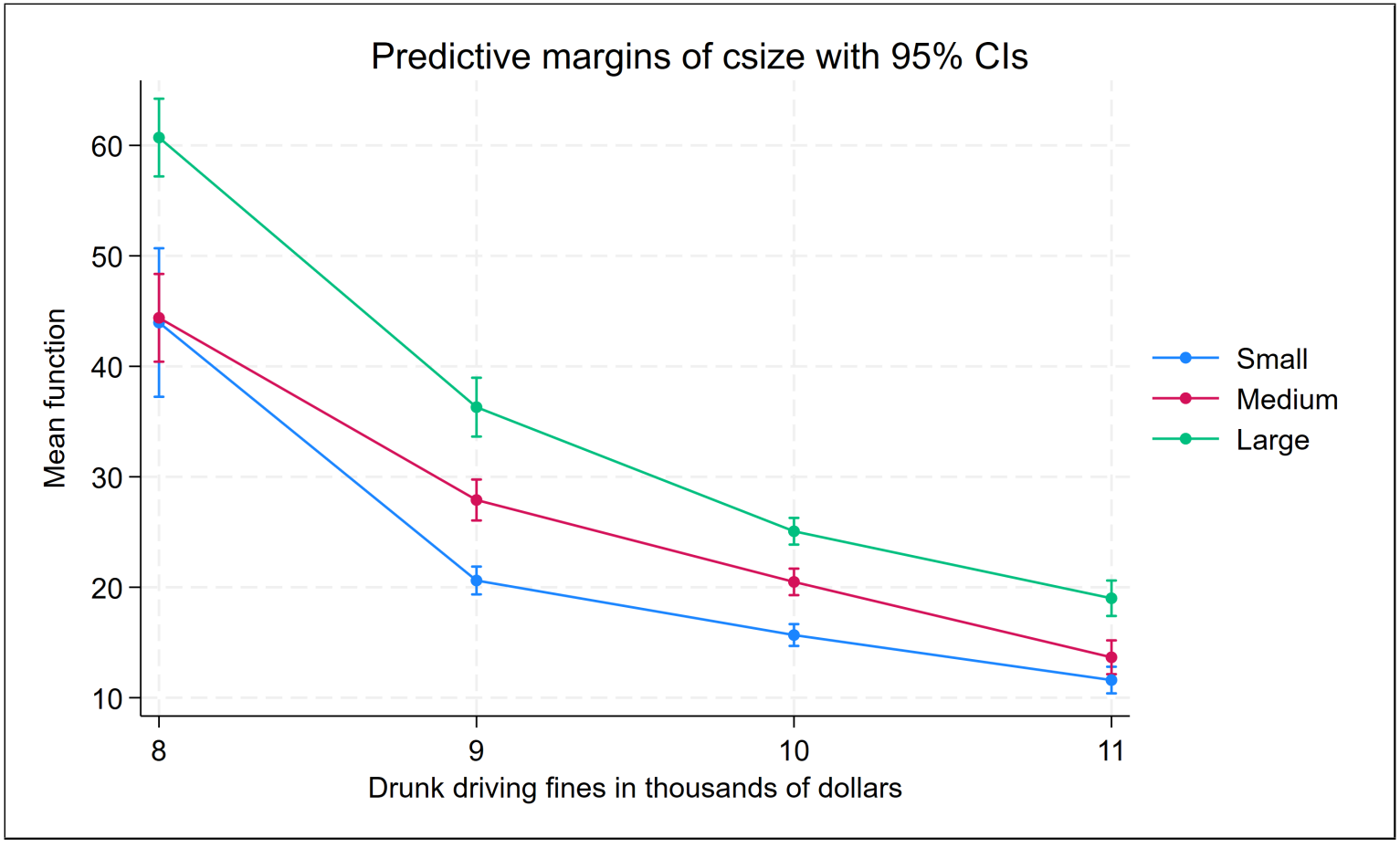
Expected citations for different levels of fines

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



The effect of fines for different levels of jurisdiction size

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



makespline

Generates a set of variables that form a **prespecified basis**.

- Piecewise polynomial spline
- B-spline

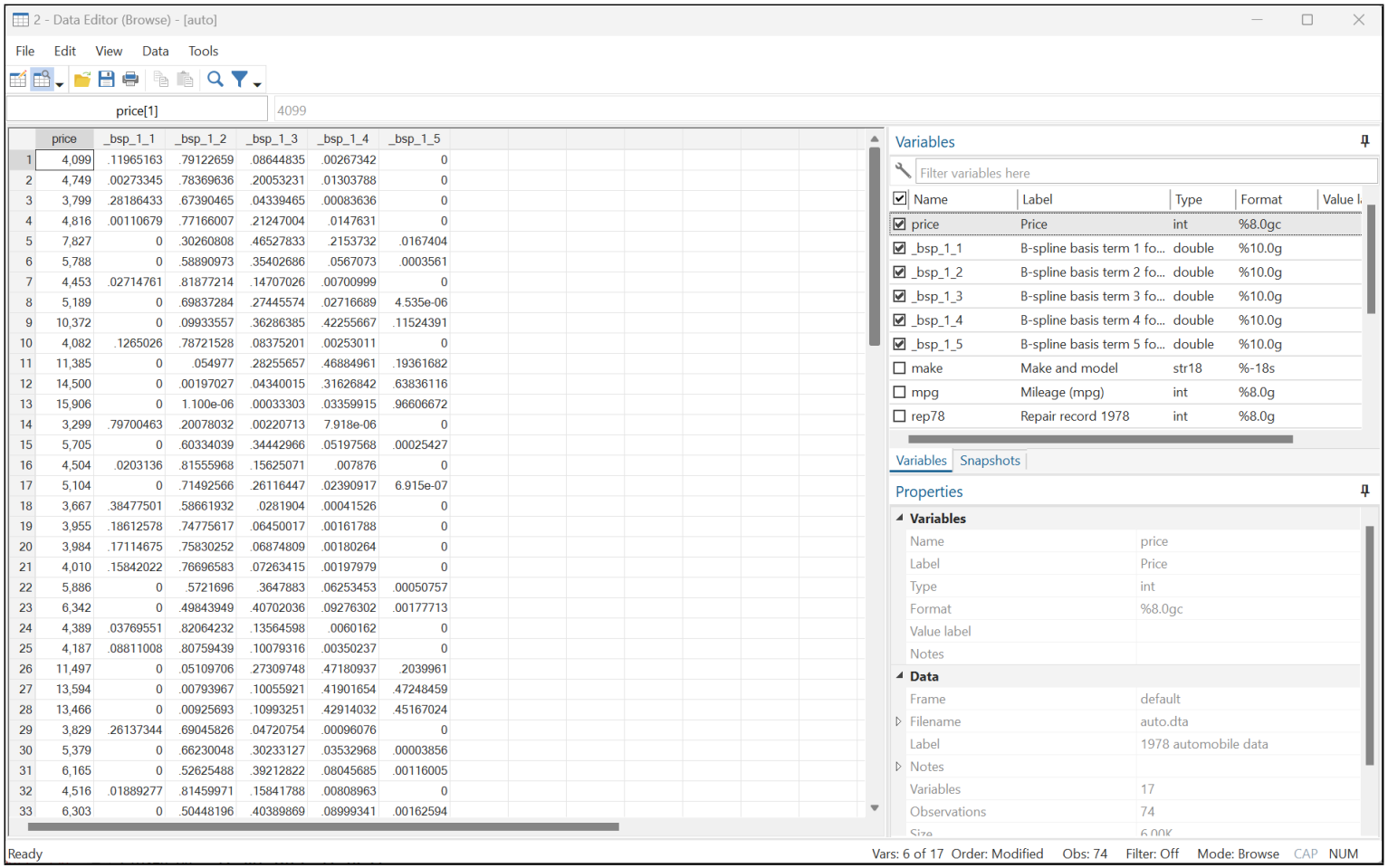
You can then use these variables **directly in regressions**. Useful in **semiparametric methods**.

Quick Start:
Generate a third-order B-spline basis from variables x_1 and x_2,

```
makespline bspline x1 x2
```

Example 11: Creating a B-spline basis

```
sysuse auto
makespline bspline price
```



Using `makespline` in Semiparametric Methods

We will use simulated data for this example:

$$y = 3x_1 + 3\sin\left(3(x_2 - x_3)\right) + \varepsilon$$

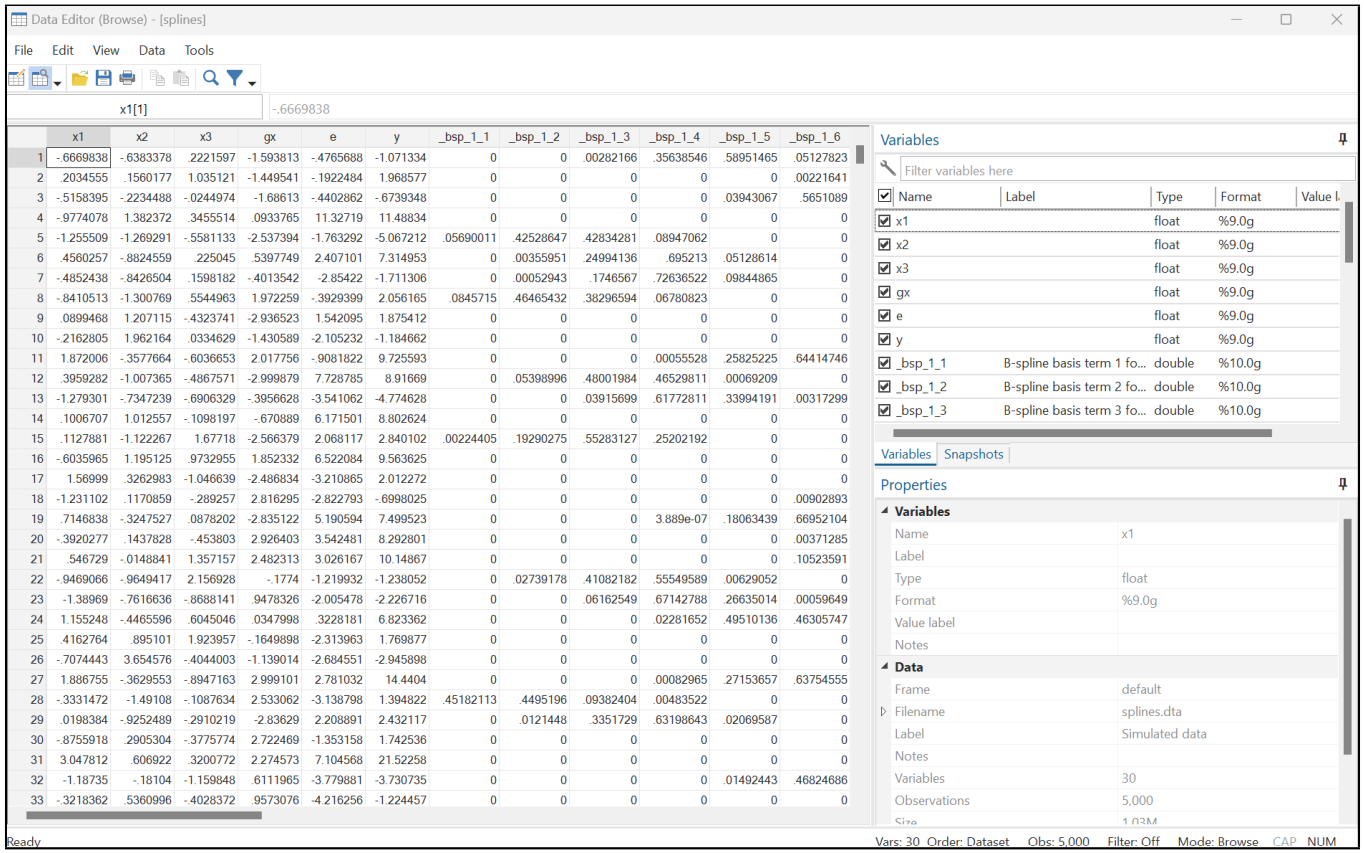
As researchers, we don't know this specific functional form. However, we assume a semiparametric model structure:

$$y = \beta_1 x_1 + g(x_2, x_3) + \varepsilon$$

Our parameter of interest is β_1. Here, $g(\cdot)$ is a nuisance parameter.

Step 1: Create the B-spline basis function

- `use https://www.stata-press.com/data/r18/splines`
- `makespline bspline x2 x3, knots(8)`



Elements of the base are stored in macro `r(regressors)`

Step 2: Use LASSO to choose elements in the base

With all the interactions we have 168 new regressors. We'll use `poregress` to select from the 168 covariates using LASSO.

- `poregress y x1, controls(‘r(regressors)’)`

```
. poregress y x1, controls(`r(regressors)')

Estimating lasso for y using plugin
Estimating lasso for x1 using plugin

Partialing-out linear model          Number of obs                =      5,000
                                     Number of controls           =        168
                                     Number of selected controls  =         19
                                     Wald chi2(1)                 =    3535.78
                                     Prob > chi2                  =     0.0000

------------------------------------------------------------------------------
             |               Robust
           y | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
          x1 |    2.951242   .049632    59.46   0.000     2.853965    3.048519
------------------------------------------------------------------------------
Note: Chi-squared test is a Wald test of the coefficients of the variables
      of interest jointly equal to zero. Lassos select controls for model
      estimation. Type lassoinfo to see number of selected variables in each
      lasso.
```

Outline

Limitations of Kernel Regression and Series Approximations

1. Computational Burden: CPU time rises fast with n and k. Problem is acuter for series approximations.

- Kernel regression.
- Option `nointeract`.

2. Curse of Dimensionality: When is k is large, we need a very big n to have informative CIs.

- Try to have a lot of data per covariate.
- Semiparametric methods.

Conclusion: Nonparametric Methods in Stata

1. Powerful tool to explore the **relationships between variables**.
   - **No assumptions on functional form** required.
   - No risk of **misspecification**.
2. Easily implementable in Stata
   - `npregress`: Estimate $g(\cdot)$, effects, and make predictions.
   - `margins`: explore $\hat{g}(\cdot)$ and ask interesting questions.
3. Be mindful of limitations:
   - **Computational cost**.
   - **Curse of dimensionality**.

Where to learn more?

1. Stata documentation:
   https://www.stata.com/features/documentation/
2. YouTube channel: https://www.youtube.com/user/statacorp
3. Send an email to our tech support team:
   tech-support@stata.com.
4. The `help` command.

Thank you!