Métodos no paramétricos en Stata

Eduardo García Echeverri

Webinar de Stata, 2023

Agenda

¿Qué es un método no paramétrico?

Método de estimación que no presupone:

- **Densidad de probabilidad** para el outcome y los regresores.
- **Forma funcional** relacionando el outcome y los regresores.

Un **método paramétrico**, requiere de estos dos supuestos.

Ejemplo

¿Cuál es el efecto de **fumar durante el embarazo** (`msmoke`) en el **peso del bebé** al nacer (`bweight`)?

- **Paramétrico:**

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

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

- **No Paramétrico:**

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

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

El Atractivo de los Métodos No Paramétricos

Ventajas:

1. Evitan problemas causados por **mala especificación**.
2. Mejoran predicciones.
3. Conocer la forma funcional no es necesario para responder nuestras preguntas de investigación.
4. Fáciles de implementar en Stata.

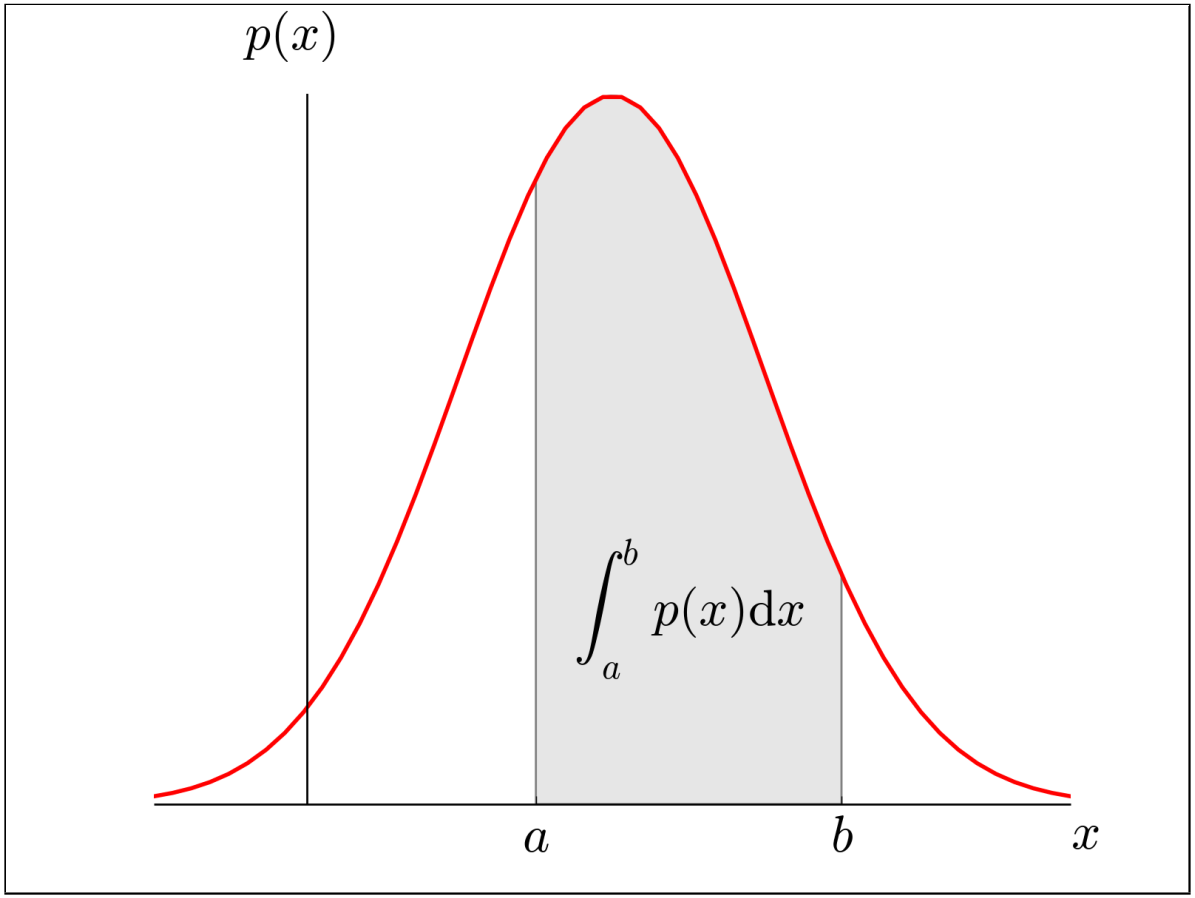
Desventajas:

1. Intensivos en datos (esp. con muchos regresores)
2. Computacionalmente costosos.

Agenda

¿Qué es una densidad de probabilidad?

Nos muestra la **distribución de una variable aleatoria**.



Fuente: deepai.org

¿Por qué son útiles?

La **densidad de probabilidad** nos muestra:

1. Qué valores son **probables/improbables**.
2. La **moda(s)** de la distribución (**picos** de la densidad)
3. El **rango** de valores que toma la variable
4. Probabilidad de **eventos extremos** (colas de la distribución)

Estimación paramétrica de densidades

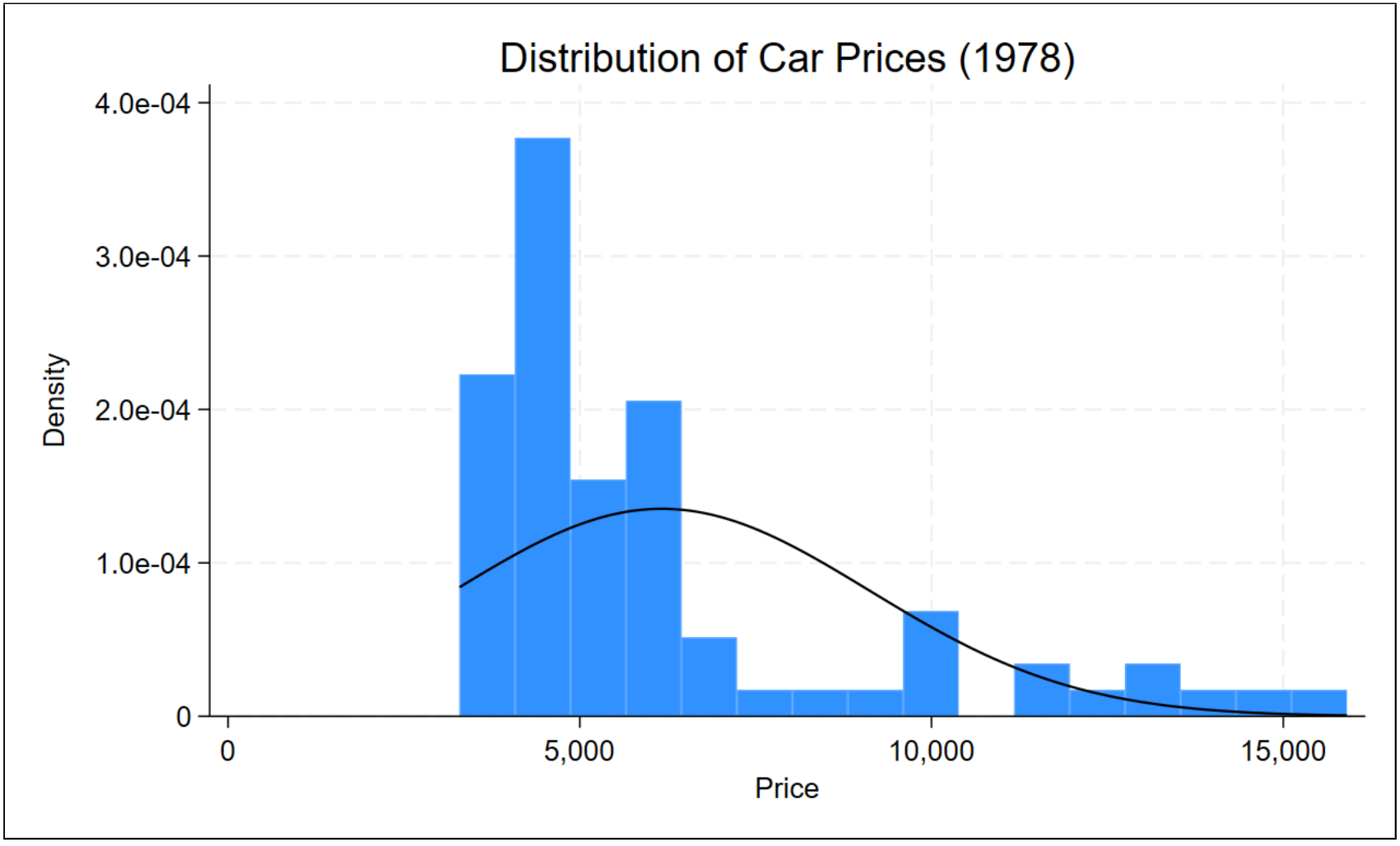
Primero, se especifica un tipo de densidad de probabilidad:

1. **Normal** (más común)
2. Log-normal
3. ...

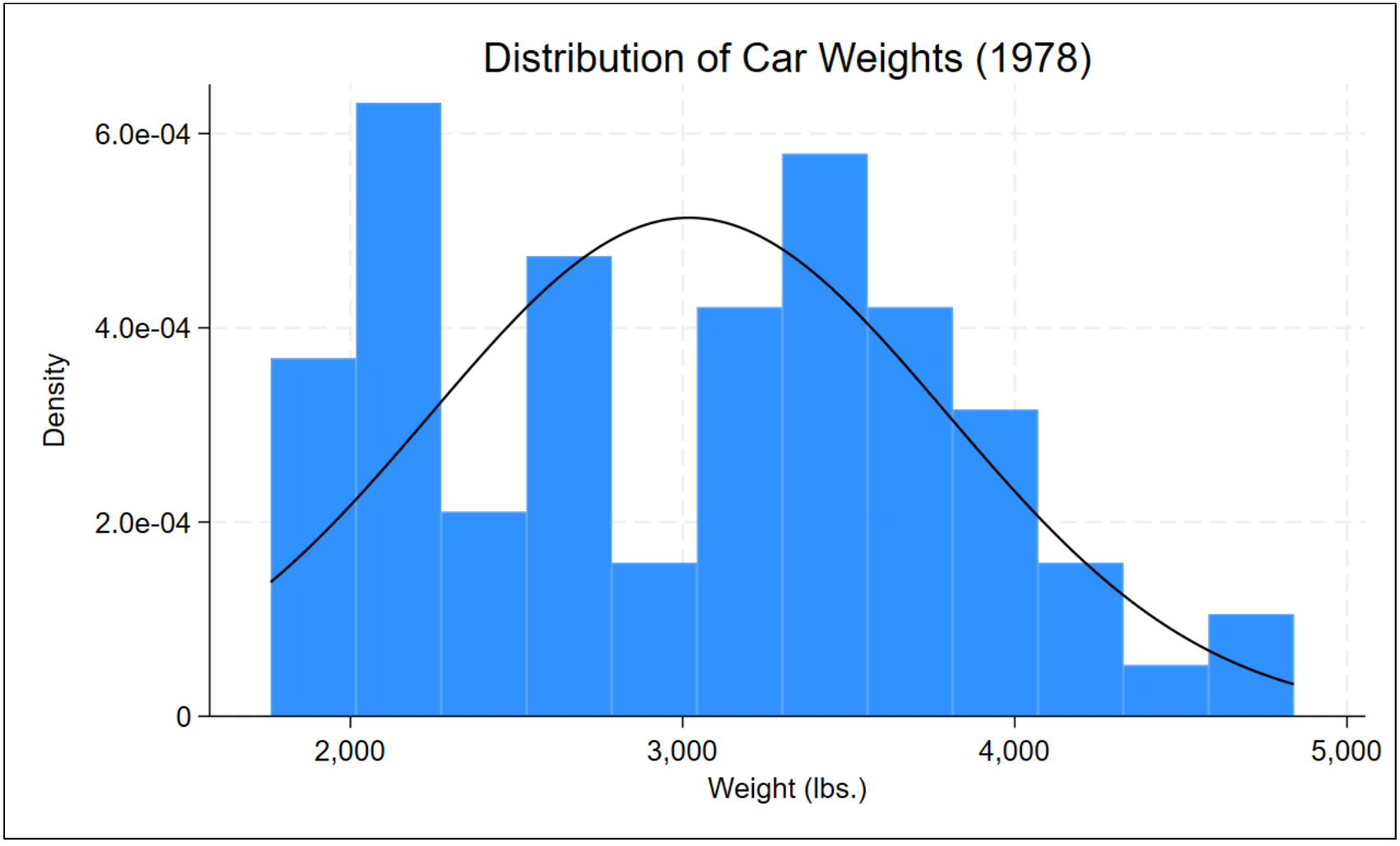
Segundo, basados en la **media y varianza muestral**, se **estima** la densidad que **mejor encaje en los datos**.

Problema: mala especificación. La normal puede ser una mala aproximación.

Normalidad puede no cumplirse – larga cola derecha



Normalidad puede no cumplirse – múltiples modas



Estimatores Kernel para densidades

Estos estimadores **no presuponen un tipo de distribución**.

Para estimar la **densidad** en el punto x:

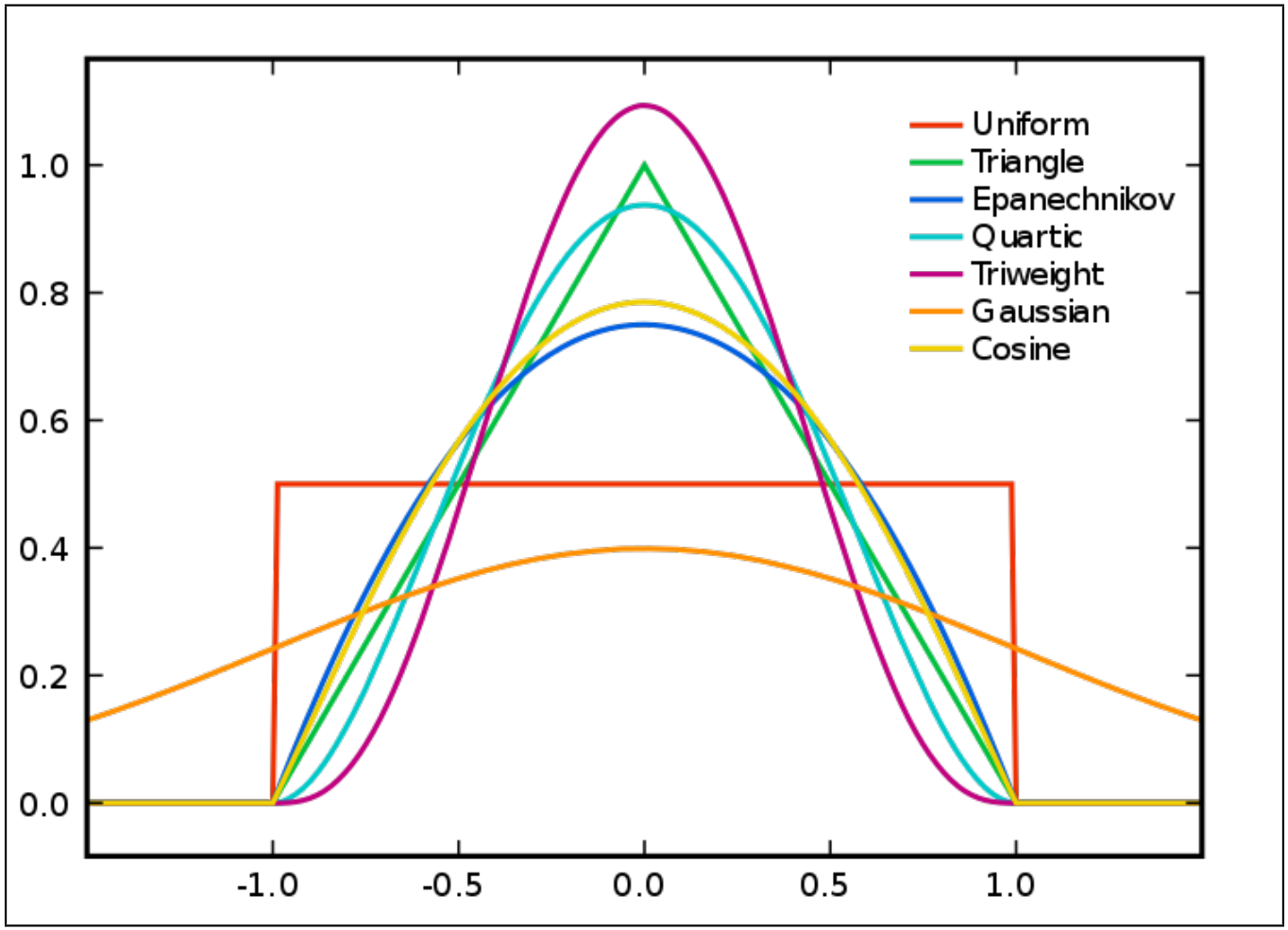
- **"Contamos"** cuántas observaciones son cercanas a x,

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

h: **Bandwidth** – número positivo (muy importante)

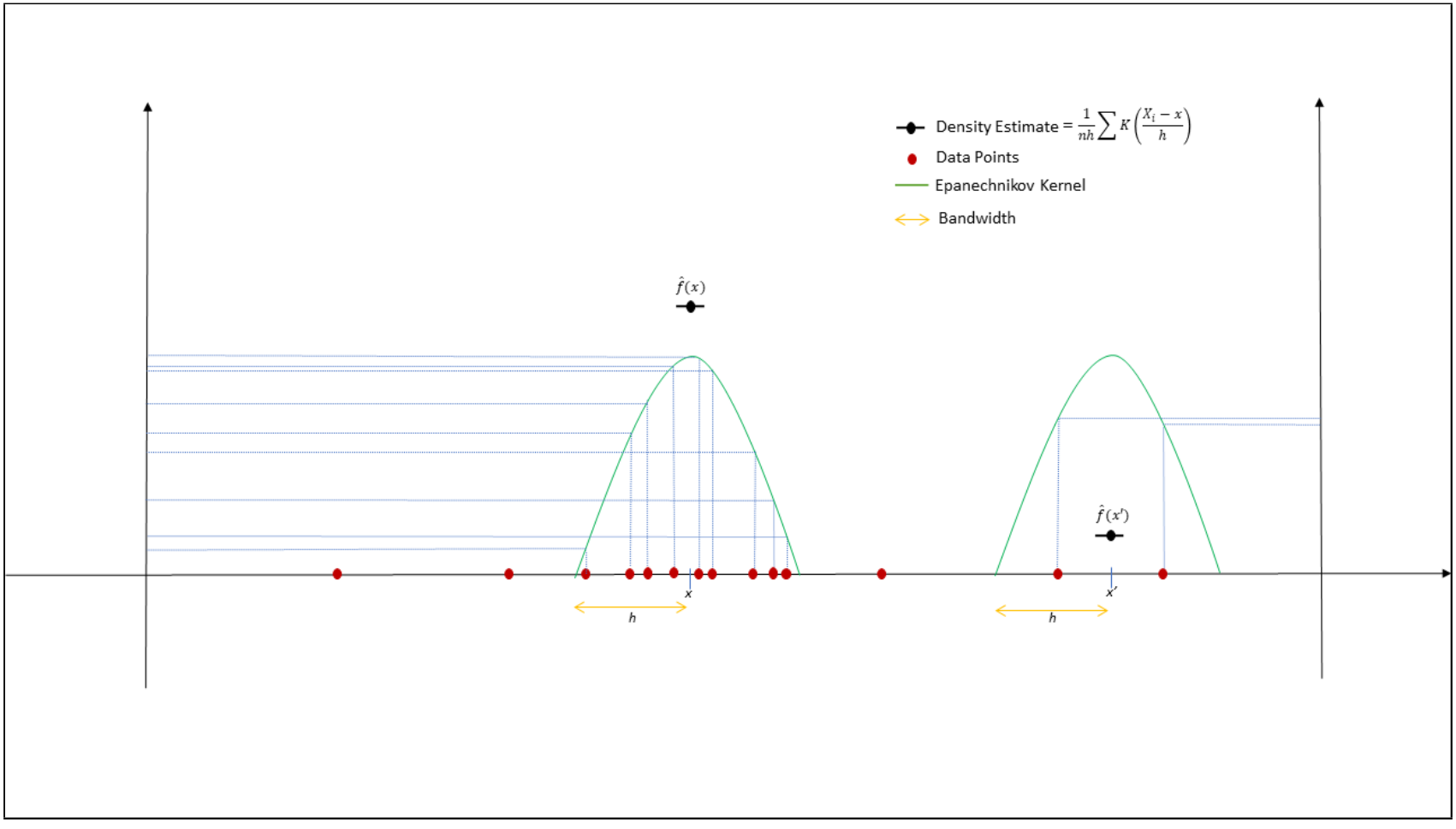
$K(\cdot)$: **Kernel** – una función ($K : \mathbb{R} \to \mathbb{R}$)

Tipos de kernel



Fuente: Wikipedia

Ilustración



Implementación en Stata

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

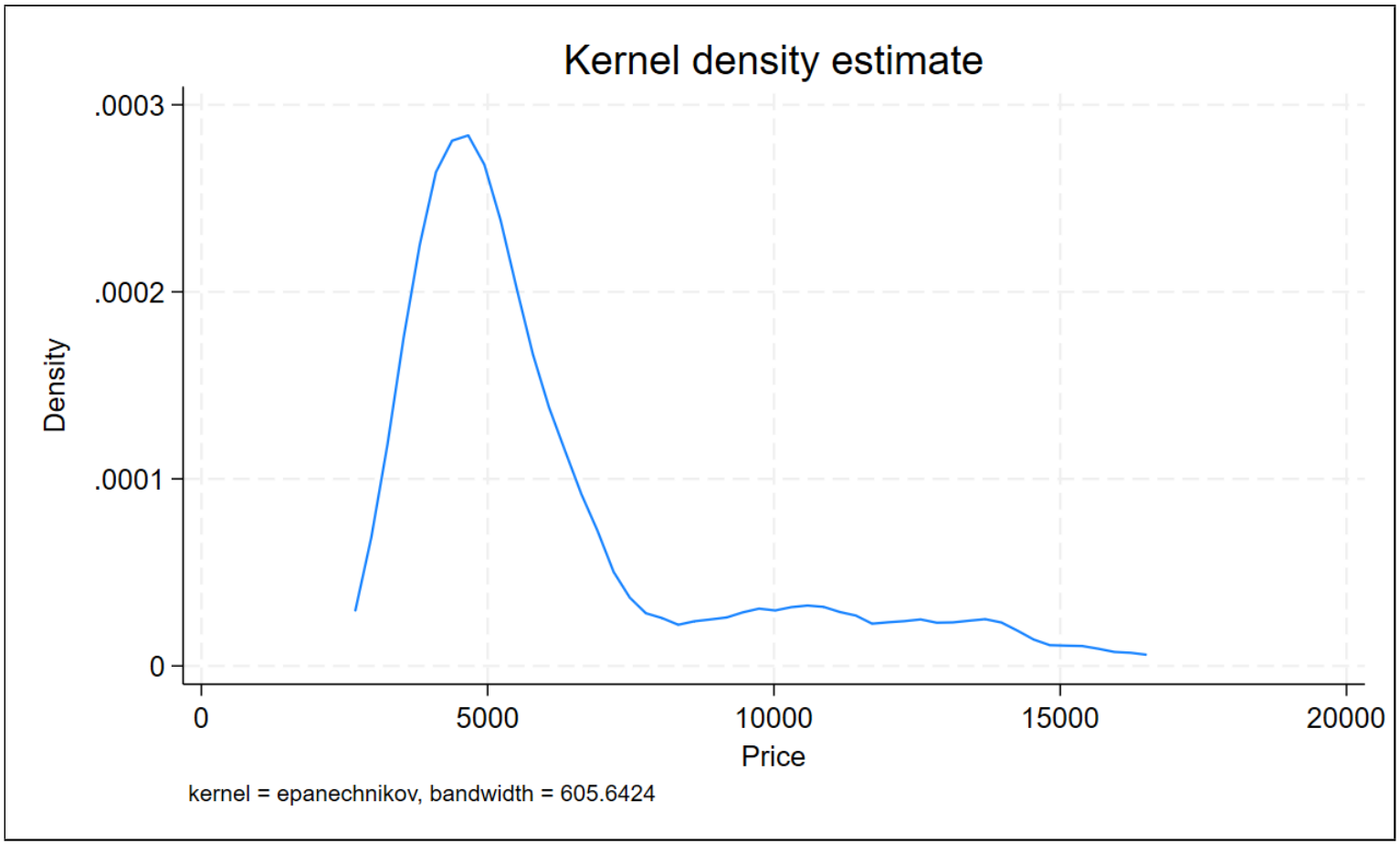
Quick Start:

Graficar la densidad de la variable $\texttt{x}_1$ usando **estimadores kernel**

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

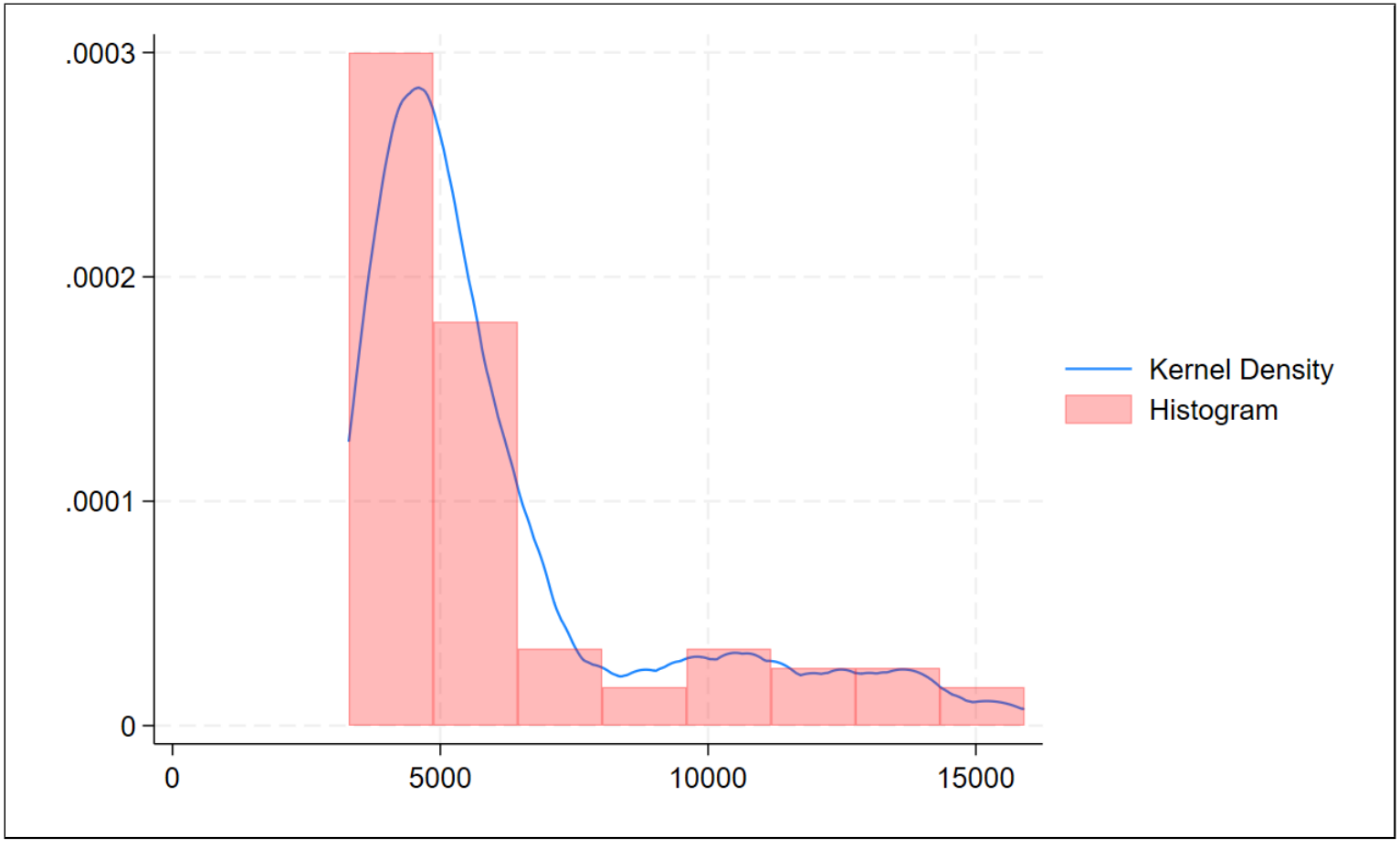
Ejemplo 1: Graficando una densidad de probabilidad

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



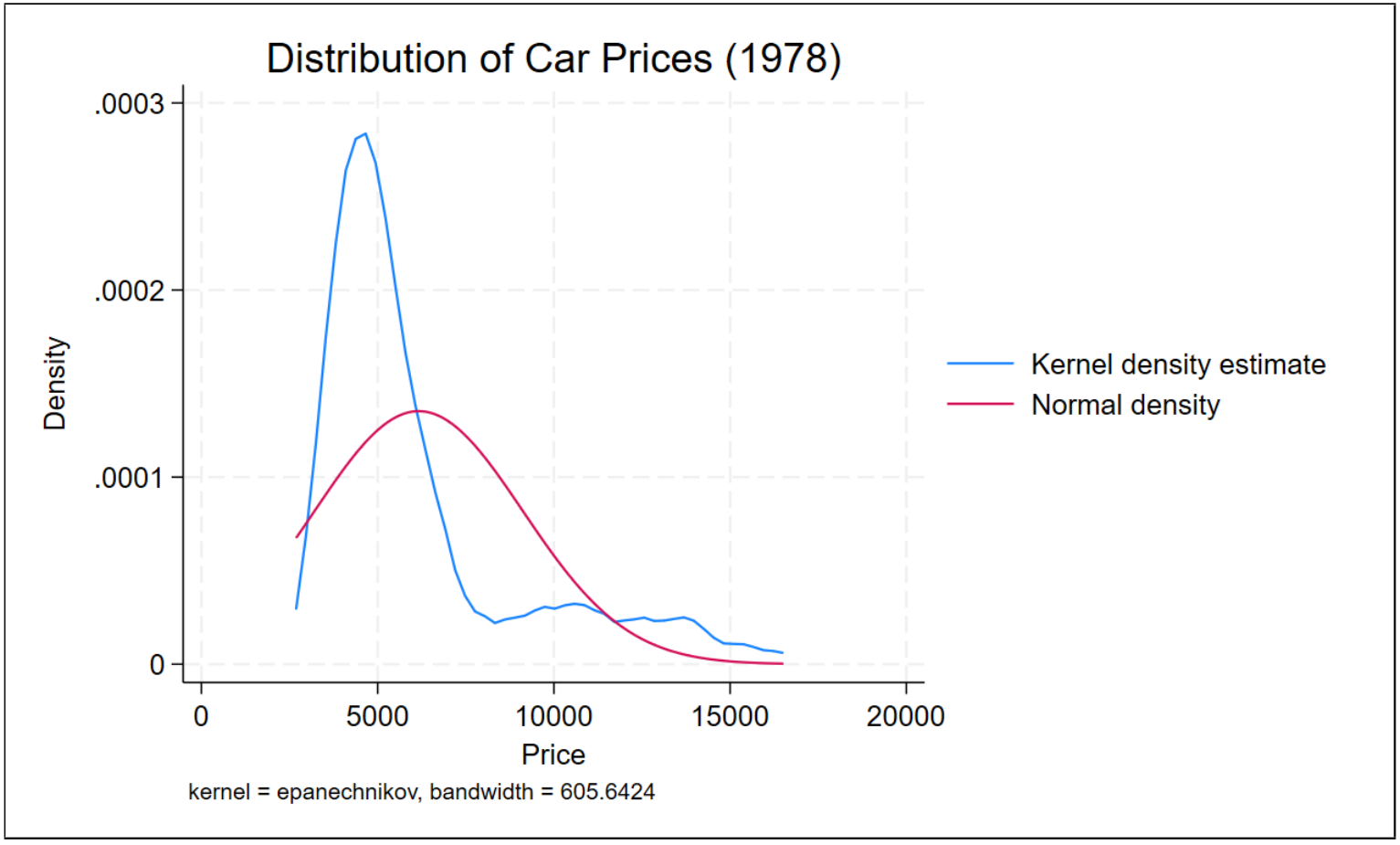
Ejemplo 1: Un mejor ajuste a los datos de precios

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



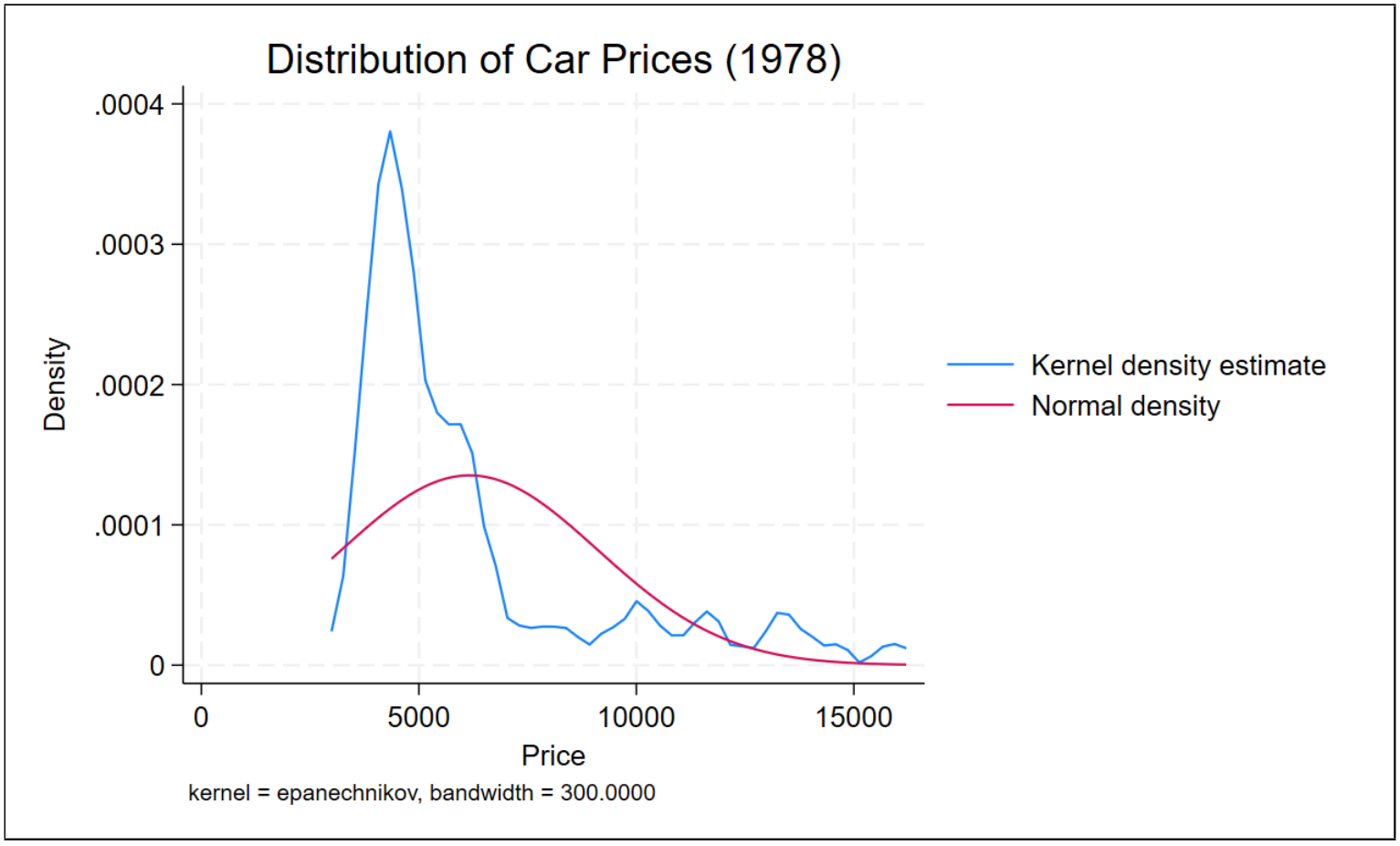
Ejemplo 2: Comparando con la distribución normal

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



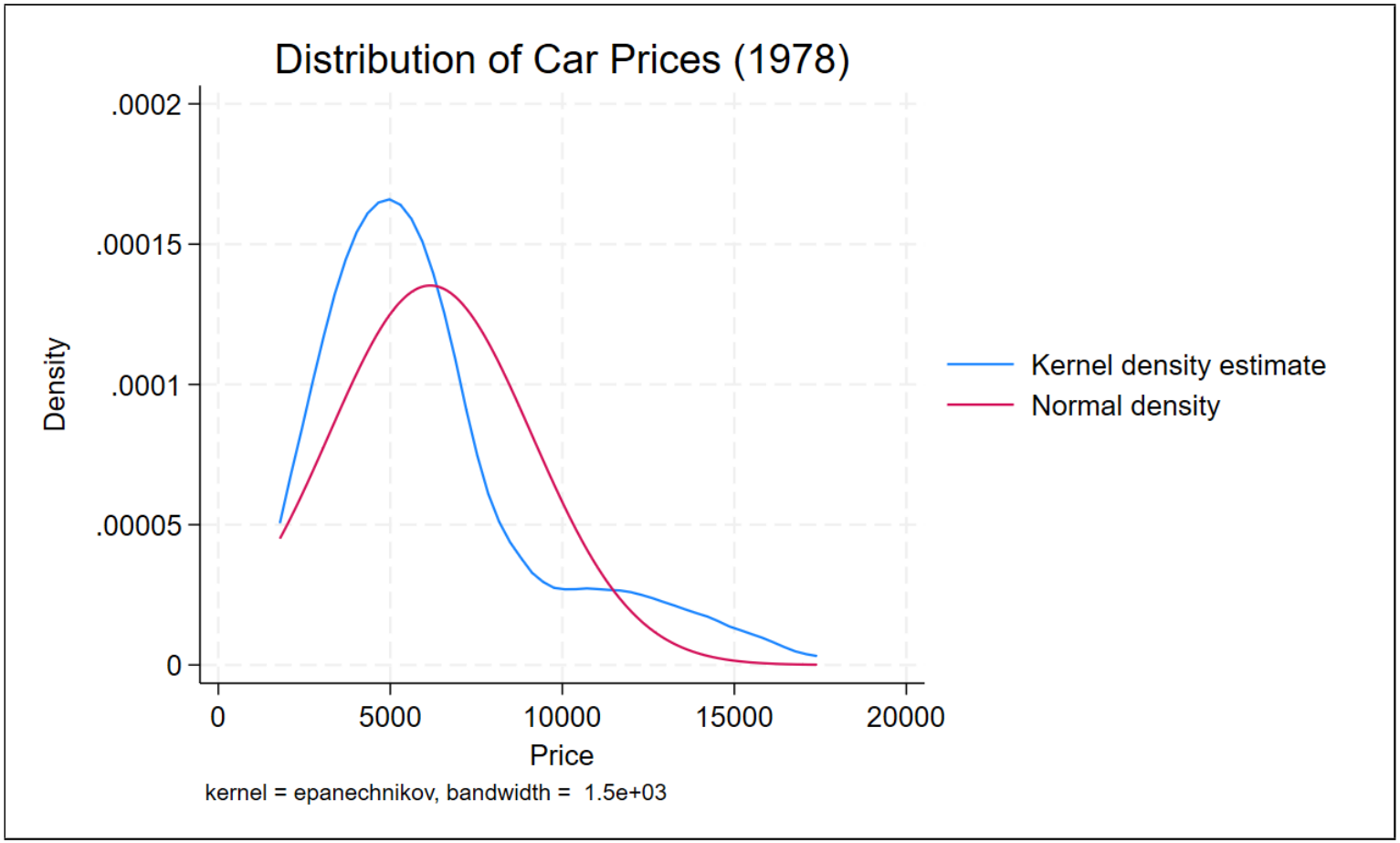
Ejemplo 3: Un bandwidth más pequeño (overfitting)

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



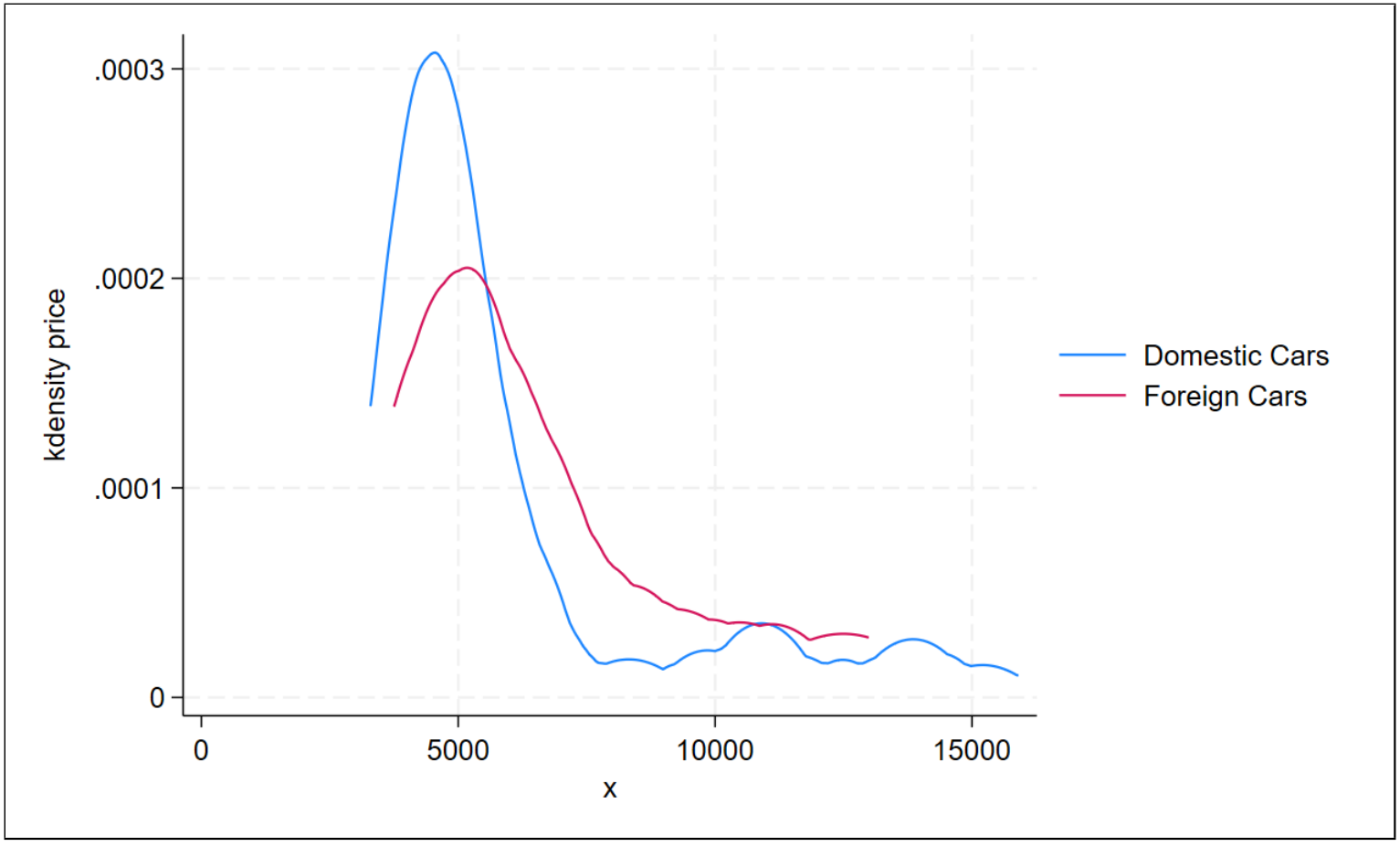
Ejemplo 4: Un bandwidth más grande (oversmoothing)

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



Ejemplo 5: Comparando distribuciones de precios

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



Opciones útiles

`kernel:` Especificar la función Kernel:

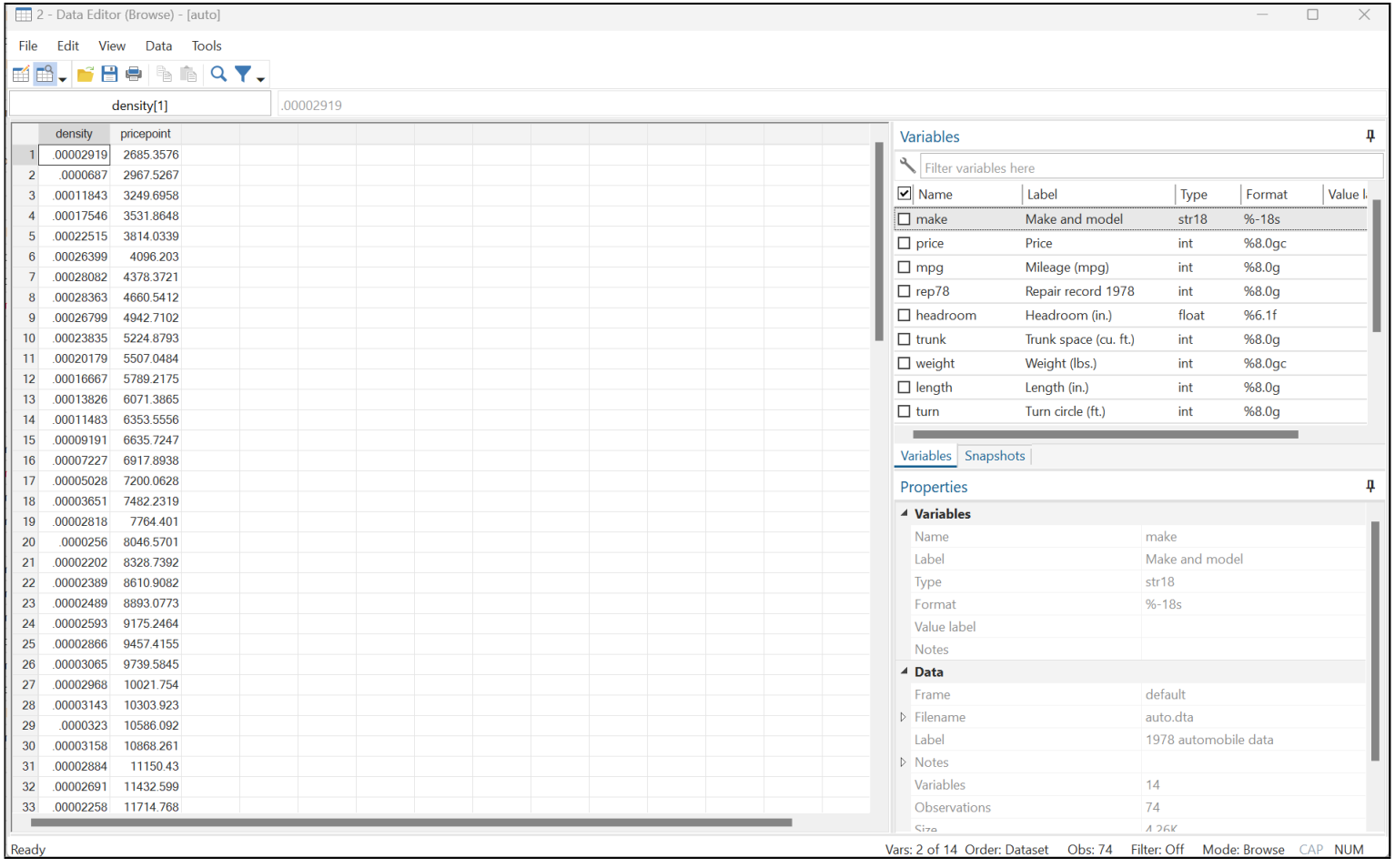
- epanechnikov; por defecto
- gaussian
- triangle...

`generate(`$newvar_1$`,` $newvar_2$`):` guardar los valores estimados en $newvar_1$ y sus respectivas densidades en $newvar_2$

`nograph:` Suprimir la gráfica.

Opción generate

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



Agenda

Esperanza condicional – Función de regresión

Objectivo:
Estimar la **esperanza condicional/función de regresión** $g(\cdot)$:

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

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

Enfoques paramétricos:

- Regresión lineal: $g(X) = x\beta$
- Probit: $g(X) = \Phi(x\beta)$
- Poisson: $g(X) = \exp(x\beta)$

Problemas por **mala especificación**.
Regresión kernel y **aproximación con series** no presuponen una forma funcional para $g(\cdot)$

Agenda

Regresión kernel – Local linear

Para estimar la **función de regresión** en el punto x:

- **"Regresión ponderada"** de y en X usando obs. cerca a 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 : **Función de regresión** estimada $\hat{g}(x)$

γ_1: **Derivada** de $g(\cdot)$ estimada en el punto x.

h: **Bandwidth** – número positivo (muy importante!)

$K(\cdot)$: **Kernel** – una función ($K : \mathbb{R} \to \mathbb{R}$)

Regresión kernel – Local constant

Para estimar la **función de regresión** en el punto x:

- **"Media ponderada"** de y usando puntos cerca de x,

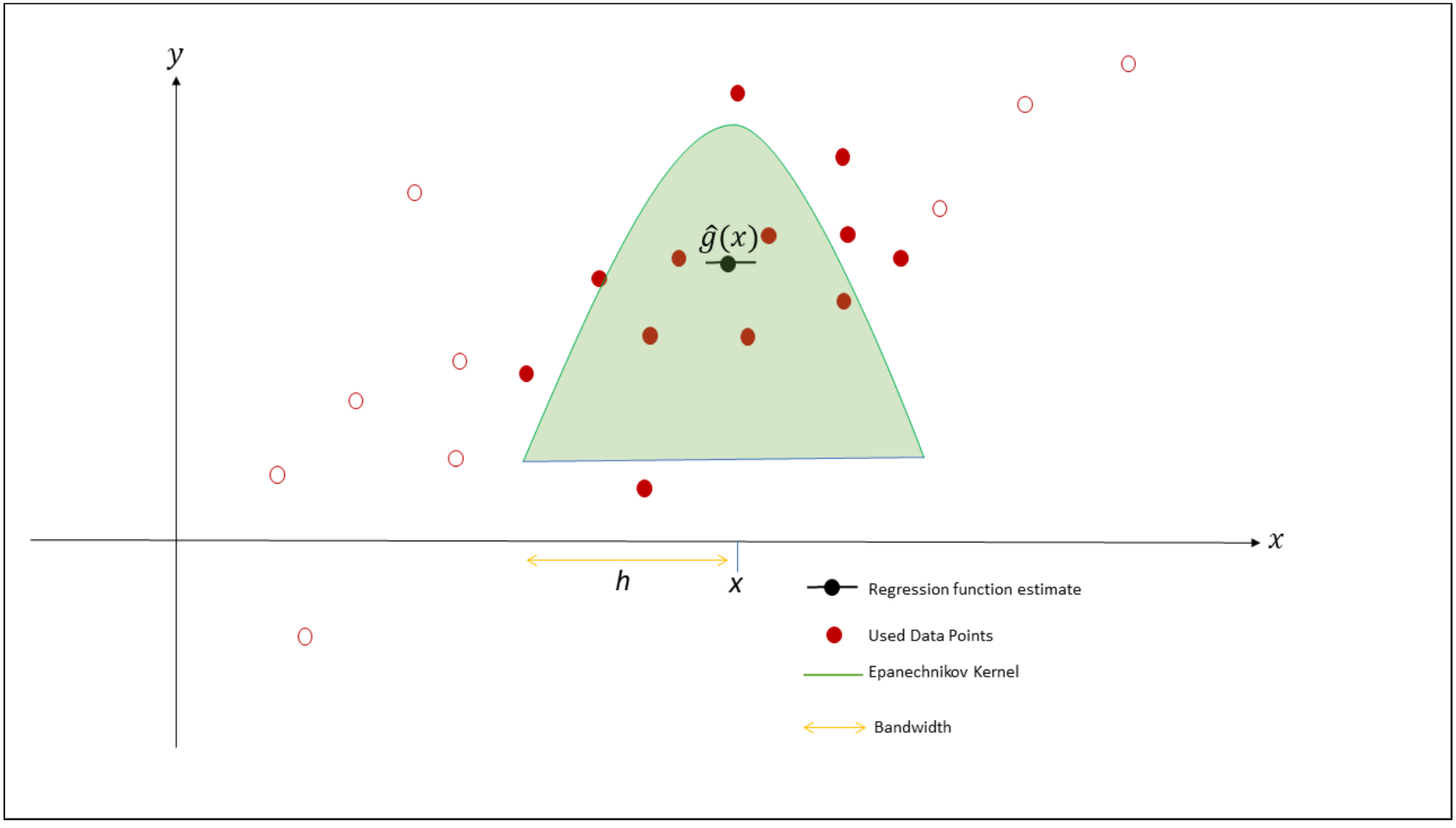
$$\min_{\gamma_0,\gamma_1} \sum_{i=1}^{n} \Big(y_i - \gamma_0\Big)^2 \cdot K\Big(\frac{x - X_i}{h}\Big)$$

γ_0 : **Función de regresión** estimada $\hat{g}(x)$

h: **Bandwidth** – número positivo (muy importante!)

$K(\cdot)$: **Kernel** – una función ($K : \mathbb{R} \to \mathbb{R}$)

Ilustración – Local Constant



Implementación en Stata

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

Quick Start:

Regresión kernel local-linear de `y` con `x` y la variable discreta `a`:

- `npregress kernel y x i.a`

Regresión kernel local-const. de `y` con `x` y la variable discreta `a`:

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

Ejemplo 7: El efecto de fumar sobre el peso de los bebés

Base de datos:

- `lbwsim2.dta`: Extracto de Cattaneo (2010) Journal of Econometrics

Outcome:

- `bweight`: **Peso del bebé** en gramos.

Treatment:

- `msmoke`: **Cigarrillos** fumados durante el embarazo

Controls:

- `mage`: Edad de la madre.
- `medu`: Años de educación de la madre.
- `alcohol`: 1 si se consumió alcohol durante el embarazo.
- `prenatal`: trimestre de la primera visita prenatal.

Ejemplo 7: Regresión kernel Local-Linear

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

Bandwidth
```

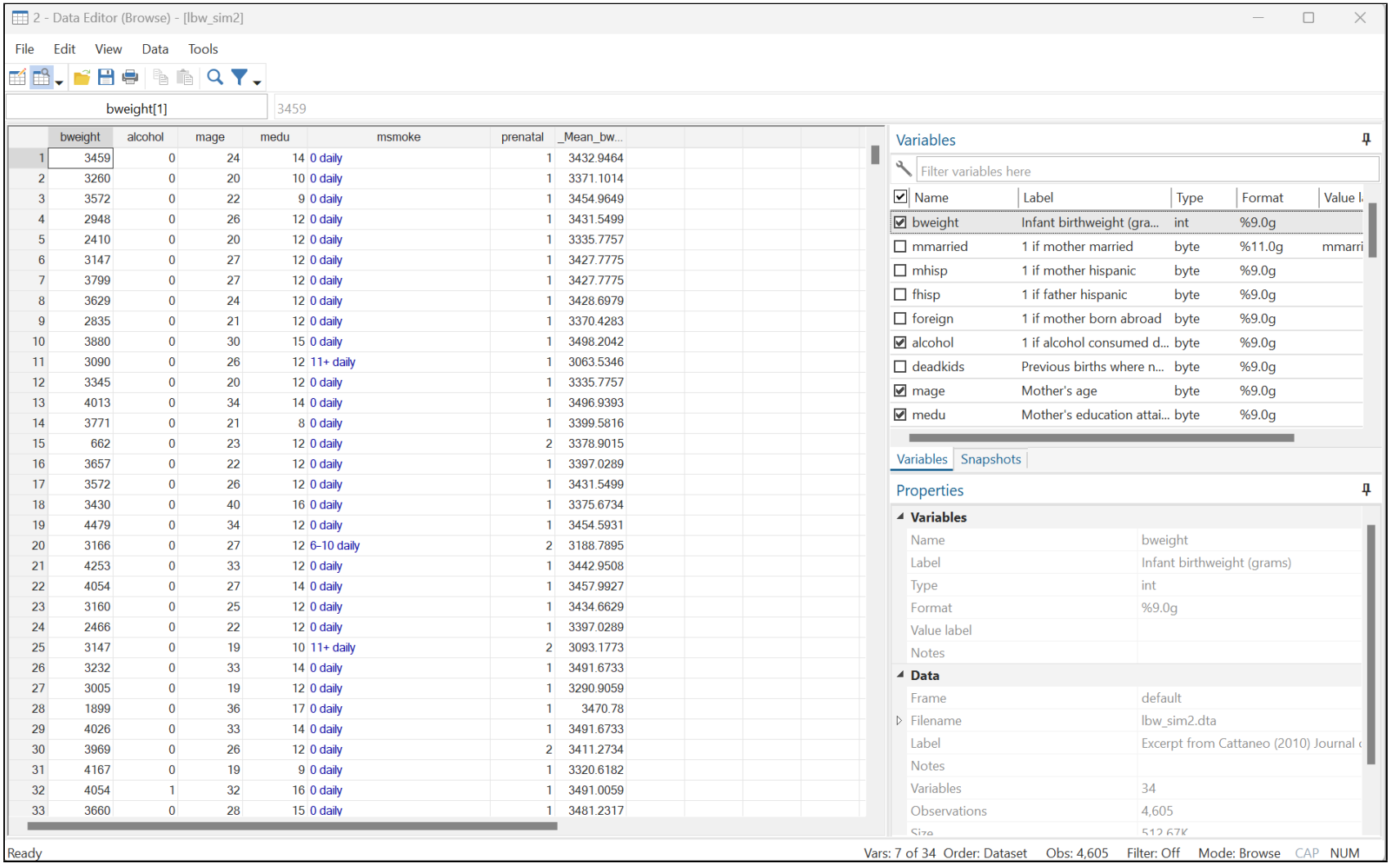
	Mean	Effect
mage	2.178108	3.09538
medu	.9284651	1.319472
msmoke	.5	.5
alcohol	.5	.5
prenatal	.5	.5

```
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().
```

Ejemplo 7: Predicciones generadas



	bweight	alcohol	mage	medu	msmoke	prenatal	_Mean_bw...
1	3459	0	24	14	0 daily	1	3432.9464
2	3260	0	20	10	0 daily	1	3371.1014
3	3572	0	22	9	0 daily	1	3454.9649
4	2948	0	26	12	0 daily	1	3431.5499
5	2410	0	20	12	0 daily	1	3335.7757
6	3147	0	27	12	0 daily	1	3427.7775
7	3799	0	27	12	0 daily	1	3427.7775
8	3629	0	24	12	0 daily	1	3428.6979
9	2835	0	21	12	0 daily	1	3370.4283
10	3880	0	30	15	0 daily	1	3498.2042
11	3090	0	26	12	11+ daily	1	3063.5346
12	3345	0	20	12	0 daily	1	3335.7757
13	4013	0	34	14	0 daily	1	3496.9393
14	3771	0	21	8	0 daily	1	3399.5816
15	662	0	23	12	0 daily	2	3378.9015
16	3657	0	22	12	0 daily	1	3397.0289
17	3572	0	26	12	0 daily	1	3431.5499
18	3430	0	40	16	0 daily	1	3375.6734
19	4479	0	34	12	0 daily	1	3454.5931
20	3166	0	27	12	6–10 daily	2	3188.7895
21	4253	0	33	12	0 daily	1	3442.9508
22	4054	0	27	14	0 daily	1	3457.9927
23	3160	0	25	12	0 daily	1	3434.6629
24	2466	0	22	12	0 daily	1	3397.0289
25	3147	0	19	10	11+ daily	2	3093.1773
26	3232	0	33	14	0 daily	1	3491.6733
27	3005	0	19	12	0 daily	1	3290.9059
28	1899	0	36	17	0 daily	1	3470.78
29	4026	0	33	14	0 daily	1	3491.6733
30	3969	0	26	12	0 daily	2	3411.2734
31	4167	0	19	9	0 daily	1	3320.6182
32	4054	1	32	16	0 daily	1	3491.0059
33	3660	0	28	15	0 daily	1	3481.2317

Ejemplo 8: Errores estándar usando un Bootstrap

```
npregress kernel bweight mage medu i.smoke i.alcohol i.prenatal, nolog vce(bootstrap, reps(40))
```

```
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
```

bweight	Observed estimate	Bootstrap std. err.	z	P>|z|	Percentile [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.
```

Ejemplo 9: Presentar los resultados gráficamente

```
. npregress kernel bweight mage, nolog

Bandwidth
```

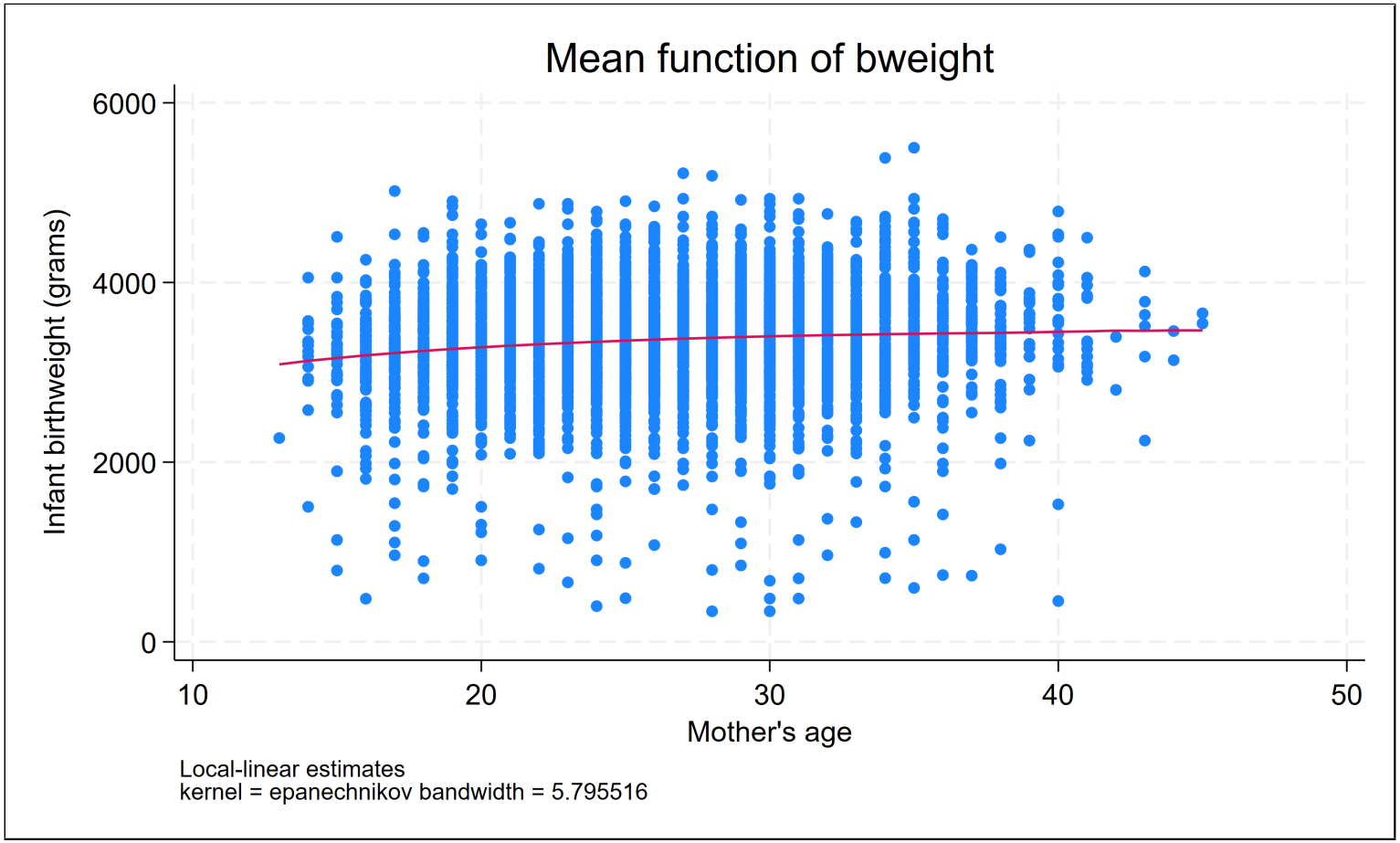
	Mean	Effect
mage	5.795516	3.695218

```
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
```

Ejemplo 9: Presentar los resultados gráficamente



Agenda

Aproximación con series

Podemos **aproximar la función** $g(\cdot)$ **usando series**. Por ejemplo, podríamos usar una **expansión de Taylor**:

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

Así, podemos **estimar la función** $g(\cdot)$ como:

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

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

Bases disponibles en Stata

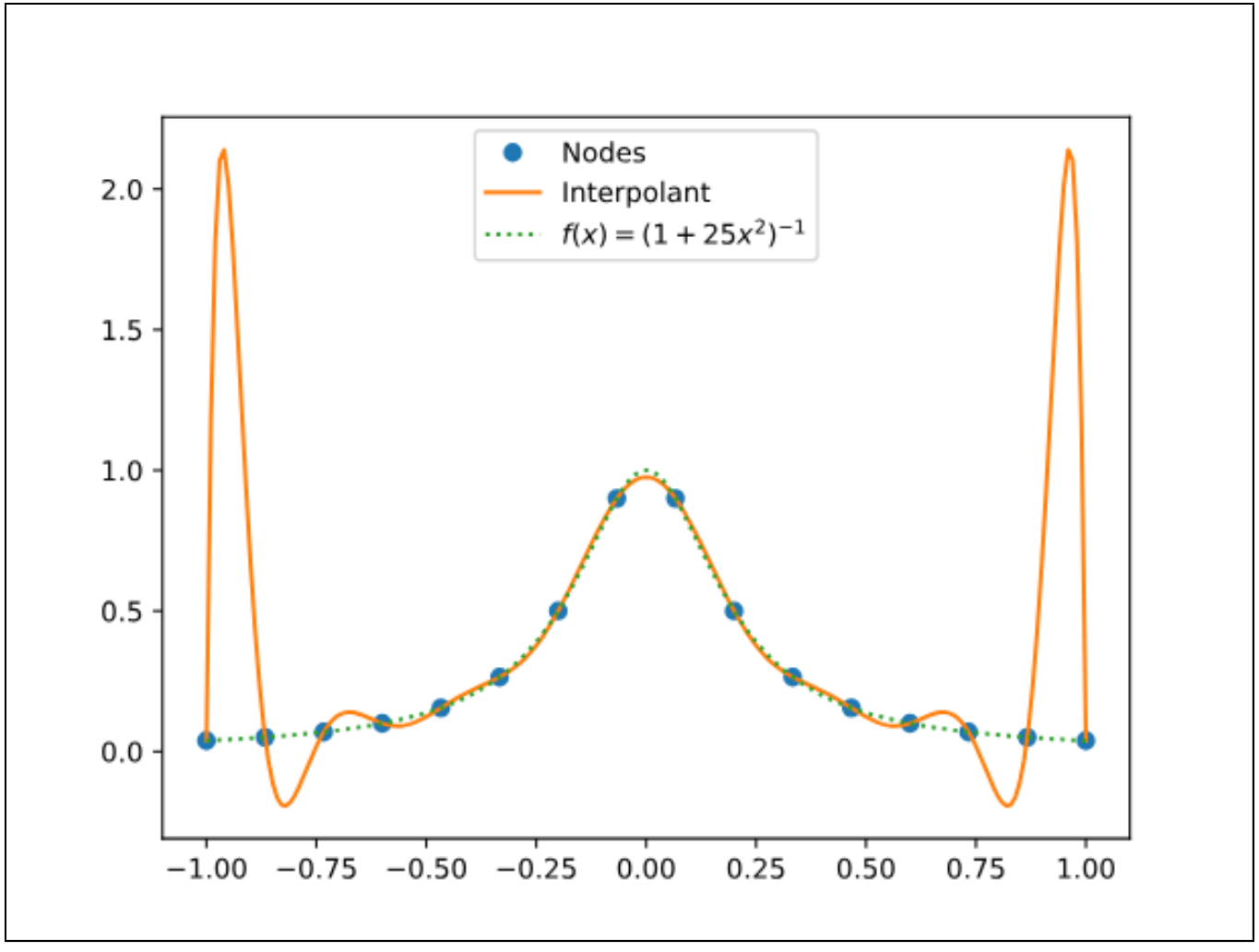
Polynomial basis:
La función $g(\cdot)$ se aproxima con un polinomio.

Piecewise polynomial spline basis:
La función $g(\cdot)$ se aproxima con un polinomio por partes.

B-Spline basis (default):
La función $g(\cdot)$ se aproxima con una función spline. Una función spline es un polinomio por partes "suavizado".

Los polinomios por partes y las splines **mitigan el fenómeno de Runge**.

Fenómeno de Runge



Fuente: John D. Cook Consulting

Implementación en Stata

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

Quick Start:

Regresión no paramétrica de `y` con `x` y la variable discreta `a` (usando una base B-spline):

- `npregress series y x i.a`

Igual que arriba pero usando una base polinomial:

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

Ejemplo 10: Efecto de las multas en las citaciones de DUI

Base de datos:

- https://www.stata-press.com/data/r18/dui (fictional data)

Outcome:

- `citations`: **Citaciones por DUI anuales** en un condado.

Tratamiento:

- `fines`: **Multa** por manejar bajo los efectos del alcohol en un condado

Controles:

- `csize`: Tamaño del condado (3 categorías)
- `college`: 1 si hay un campus universitario en el condado.

Regresión no paramétrica en Stata usando series

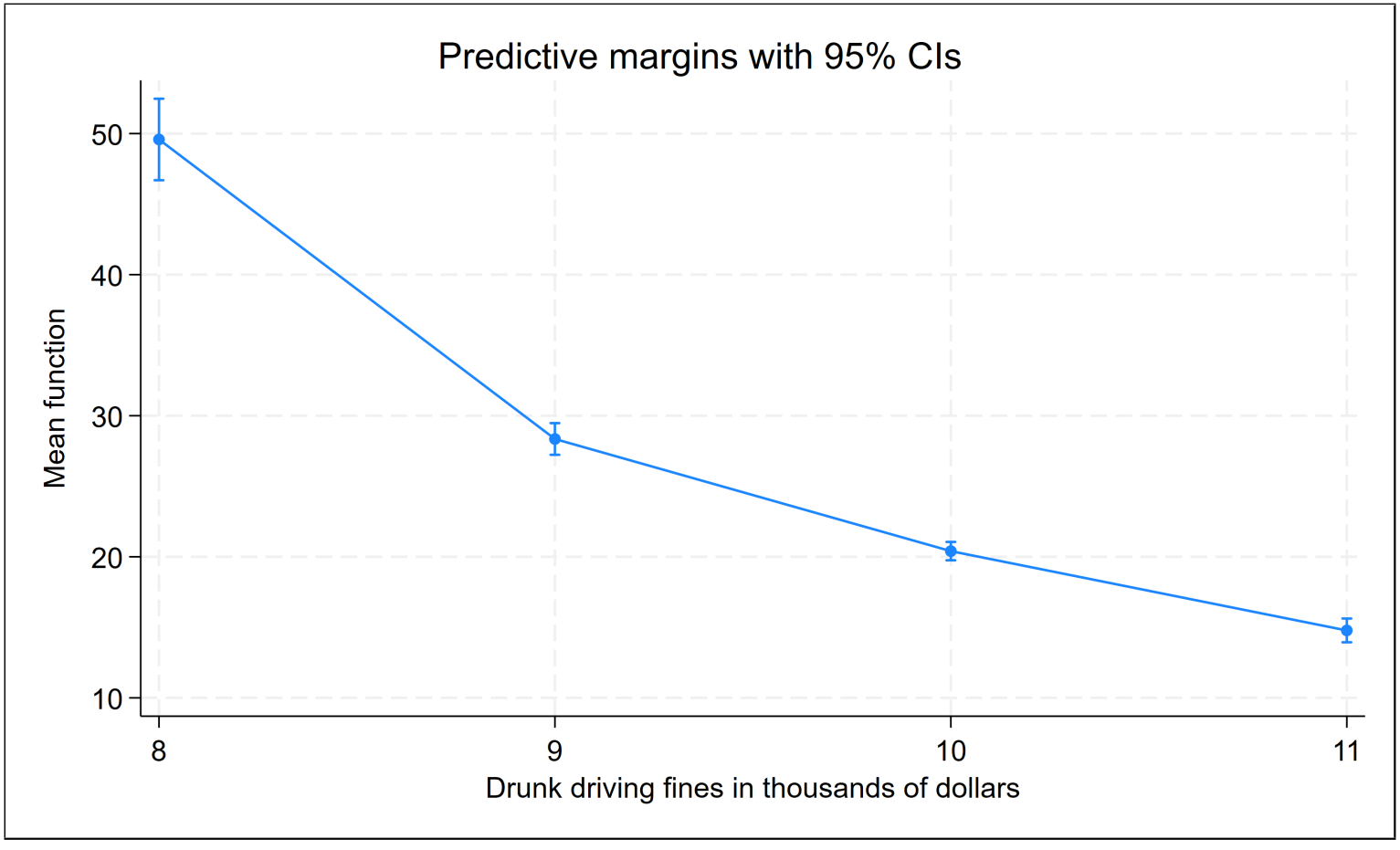
```
. npregress series citations fines i.csize i.college

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.
```

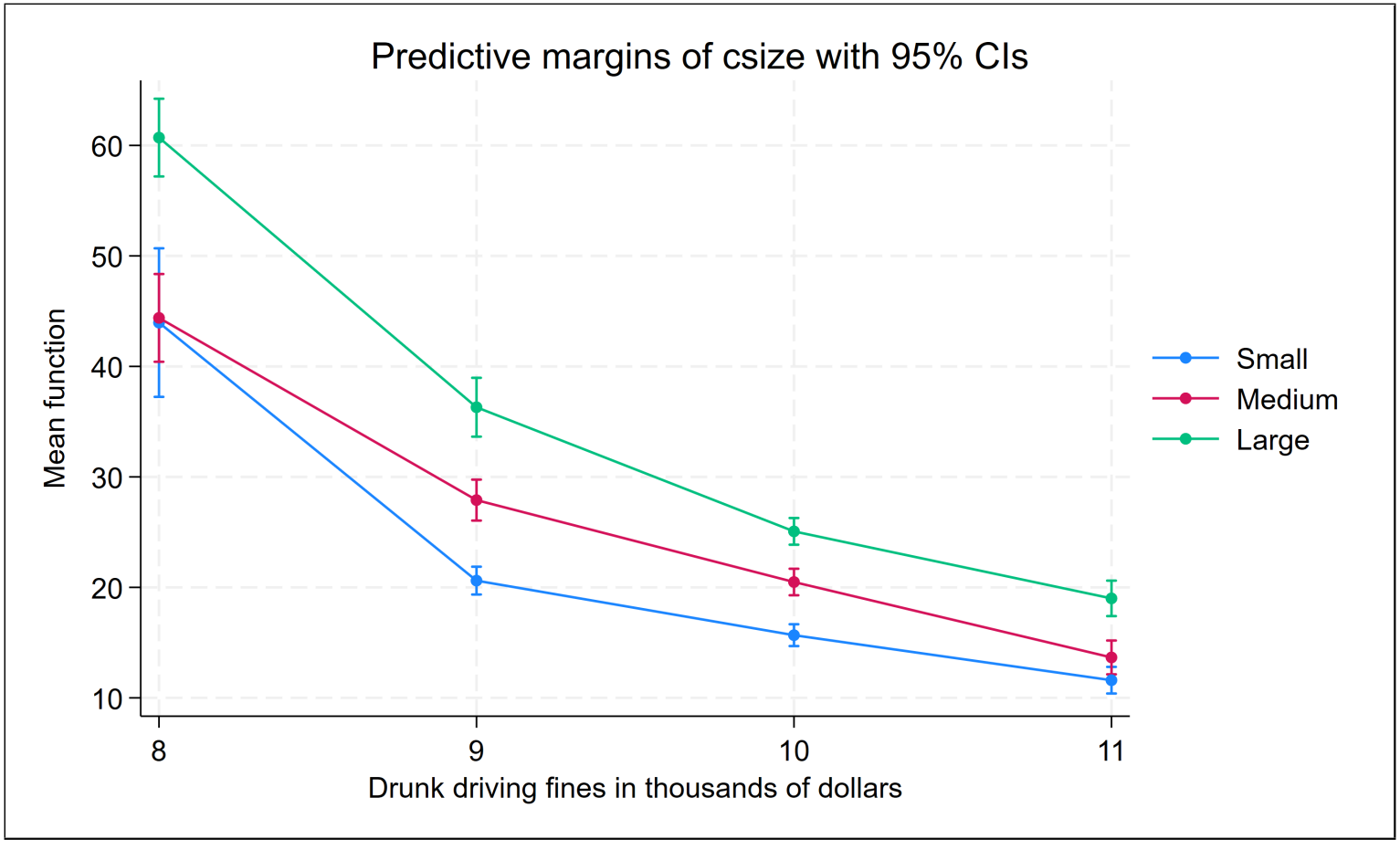
Citaciones predichas para distintos niveles de multas

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



El efecto de las multas según tamaño del condado

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



makespline

Genera un conjunto de variables que constituyen una **base predeterminada**.

- Piecewise polynomial spline
- B-spline

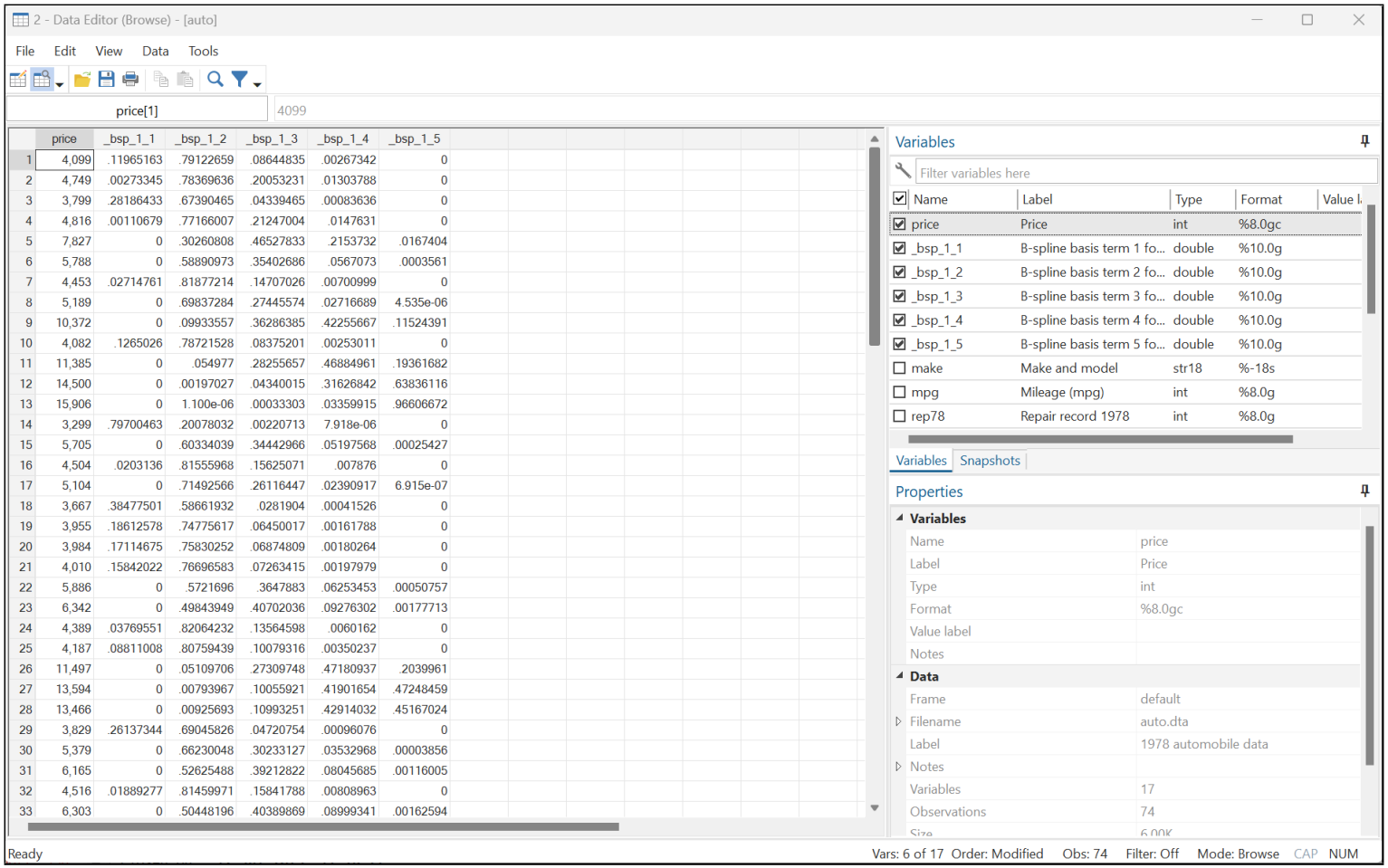
Puedes usar estas variables **directamente en tus regresiones**.
Útil para implementar **métodos semiparamétricos**.

Quick Start:
Generar una base B-spline de tercer order a partir de las variables x_1 y x_2, con nodos en las medianas de cada variable

```
makespline bspline x1 x2
```

Ejemplo 11: Creando una base B-spline

```
sysuse auto
makespline bspline price
```



Métodos semiparamétricos usando `makespline`

Usaremos **datos simulados** para este ejemplo:

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

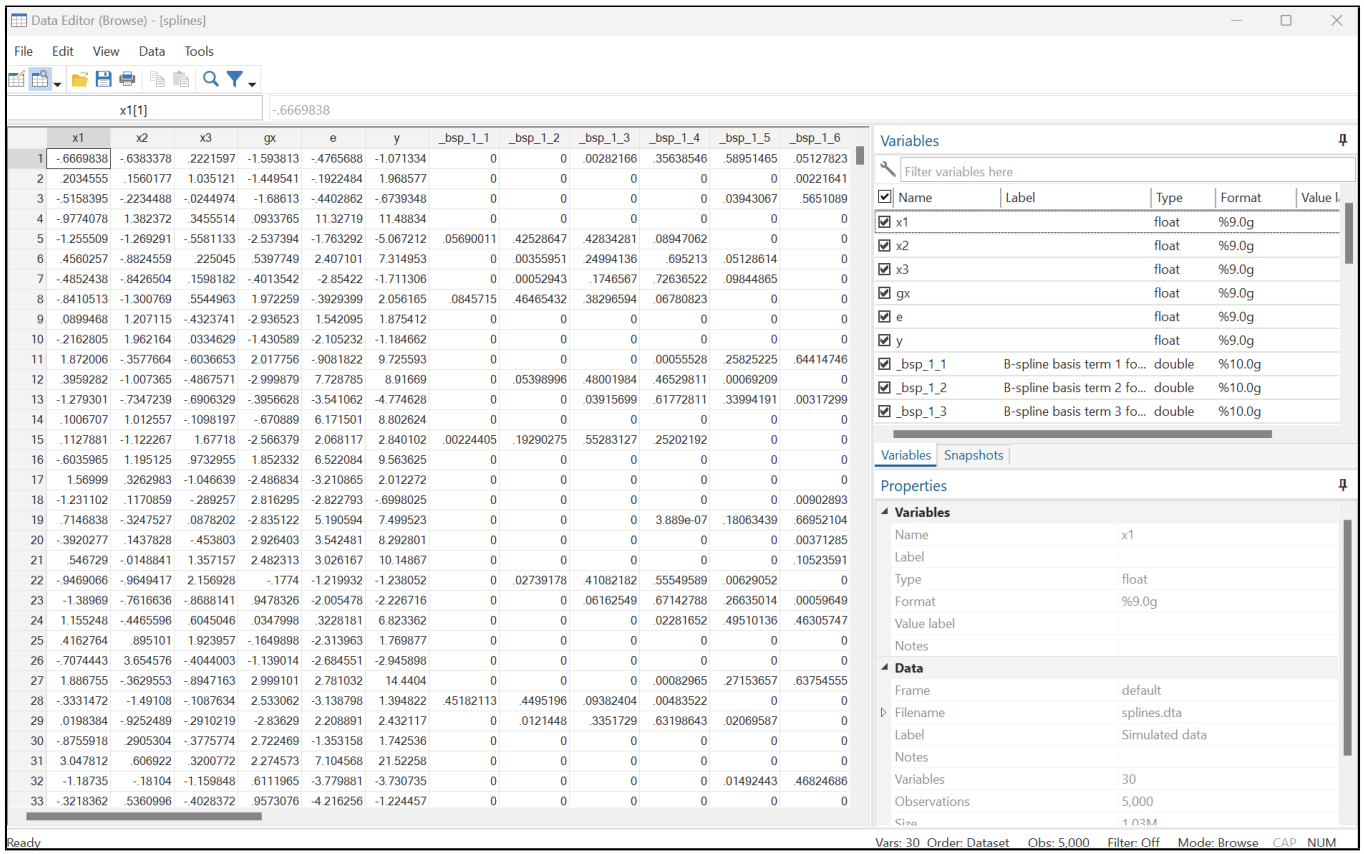
Como investigadores, **no conocemos** la forma funcional específica. No obstante, suponemos un **modelo semiparamétrico**:

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

Nuestro **parámetro de interés** es β_1. Aquí, $g(\cdot)$ es un **parámetro secundario**.

Paso 1: Crear la base B-Spline

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



	x1	x2	x3	gx	e	y	_bsp_1_1	_bsp_1_2	_bsp_1_3	_bsp_1_4	_bsp_1_5	_bsp_1_6
1	-.6669838	-.6383378	.2221597	-1.593813	-.4765688	-1.071334	0	0	.00282166	.35638546	.58951465	.05127823
2	.2034555	.1560177	1.035121	-1.449541	-.1922484	1.968577	0	0	0	0	0	.00221641
3	-.5158395	-.2234488	-.0244974	-1.68613	-.4402862	-.6739348	0	0	0	0	.03943067	.5651089
4	-.9774078	1.382372	.3455514	.0933765	11.32719	11.48834	0	0	0	0	0	0
5	-1.255509	-1.269291	-.5581133	-2.537394	-1.763292	-5.067212	.05690011	.42528647	.42834281	.08947062	0	0
6	.4560257	-.8824559	.225045	.5397749	2.407101	7.314953	0	.00355951	.24994136	.695213	.05128614	0
7	-.4852438	-.8426504	.1598182	-.4013542	-2.85422	-1.711306	0	.00052943	.1746567	.72636522	.09844865	0
8	-.8410513	-1.300769	.5544963	1.972259	-.3929399	2.056165	.0845715	.46465432	.38296594	.06780823	0	0
9	.0899468	1.207115	-.4323741	-2.936523	1.542095	1.875412	0	0	0	0	0	0
10	-.2162805	1.962164	.0334629	-1.430589	-2.105232	-1.184662	0	0	0	0	0	0
11	1.872006	-.3577664	-.6036653	2.017756	-.9081822	9.725593	0	0	0	.00055528	.25825225	.64414746
12	.3959282	-1.007365	-.4867571	-2.999879	7.728785	8.91669	0	.05398996	.48001984	.46529811	.00069209	0
13	-1.279301	-.7347239	-.6906329	-.3956628	-3.541062	-4.774628	0	0	.03915699	.61772811	.33994191	.00317299
14	.1006707	1.012557	-.1098197	-.670889	6.171501	8.802624	0	0	0	0	0	0
15	.1127881	-1.122267	1.67718	-2.566379	2.068117	2.840102	.00224405	.19290275	.55283127	.25202192	0	0
16	-.6035965	1.195125	.9732955	1.852332	6.522084	9.563625	0	0	0	0	0	0
17	1.56999	.3262983	-1.046639	-2.486834	-3.210865	2.012272	0	0	0	0	0	0
18	-1.231102	.1170859	-.289257	2.816295	-2.822793	-.6998025	0	0	0	0	0	.00902893
19	.7146838	-.3247527	.0878202	-2.835122	5.190594	7.499523	0	0	0	3.889e-07	.18063439	.66952104
20	-.3920277	.1437828	-.453803	2.926403	3.542481	8.292801	0	0	0	0	0	.00371285
21	.546729	-.0148841	1.357157	2.482313	3.026167	10.14867	0	0	0	0	0	.10523591
22	-.9469066	-.9649417	2.156928	-.1774	-1.219932	-1.238052	0	.02739178	.41082182	.55549589	.00629052	0
23	-1.38969	-.7616636	-.8688141	.9478326	-2.005478	-2.226716	0	0	.06162549	.67142788	.26635014	.00059649
24	1.155248	-.4465596	.6045046	.0347998	.3228181	6.823362	0	0	0	.02281652	.49510136	.46305747
25	.4162764	.895101	1.923957	-.1649898	-2.313963	1.769877	0	0	0	0	0	0
26	-.7074443	3.654576	-.4044003	-1.139014	-2.684551	-2.945898	0	0	0	0	0	0
27	1.886755	-.3629553	-.8947163	2.999101	2.781032	14.4404	0	0	0	.00082965	.27153657	.63754555
28	-.3331472	-1.49108	-.1087634	2.533062	-3.138798	1.394822	.45182113	.4495196	.09382404	.00483522	0	0
29	.0198384	-.9252489	-.2910219	-2.83629	2.208891	2.432117	0	.0121448	.3351729	.63198643	.02069587	0
30	-.8755918	.2905304	-.3775774	2.722469	-1.353158	1.742536	0	0	0	0	0	0
31	3.047812	.606922	.3200772	2.274573	7.104568	21.52258	0	0	0	0	0	0
32	-1.18735	-.18104	-1.159848	.6111965	-3.779881	-3.730735	0	0	0	0	.01492443	.46824686
33	-.3218362	.5360996	-.4028372	.9573076	-4.216256	-1.224457	0	0	0	0	0	0

Los elementos de la base **se guardan en** `r(regressors)`

Paso 2: Usar LASSO para escoger los elementos de la base

Con todas las interacciones tendríamos **168 regresores**. Usamos `poregress` para seleccionar de entre estos usando **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.
```

Agenda

Limitaciones de regresión kernel y aproximación con series

1. Costo computacional: Tiempo CPU incrementa rápidamente con n y k. Problema es más agudo en aproximación con series.

- Regresión kernel
- Opción `nointeract`.

2. 'Maldición de la dimensionalidad': Cuando k es grande, n debe ser muy grande para que los IC sean informativos.

- Intentar tener bastantes observaciones por regresor.
- Métodos semiparamétricos.

Conclusión: Métodos no paramétricos en Stata

1. Herramienta poderosa para explorar las **relaciones entre variables**.
   - No requieren **supuestos de forma funcional**.
   - No hay riesgo de **mala especificación**.
2. Fáciles de implementar en Stata
   - `npregress:` Estimar $g(\cdot)$, efectos, y hacer predicciones.
   - `margins:` explorar $\hat{g}(\cdot)$ y responder preguntas interesantes.
3. Cuidado con las limitaciones:
   - **Costo computacional**.
   - **'Maldición de la dimensionalidad'**.

¿Dónde aprender más?

1. Documentación de Stata: https://www.stata.com/features/documentation/
2. Canal de YouTube: https://www.youtube.com/user/statacorp
3. Envía un correo a nuestro soporte técnico: tech-support@stata.com.
4. El comando `help`.

¡Gracias!