# Topics in Applied Econometrics for Public Policy

Master in Economics of Public Policy, BSE

## Handout 2: Introduction to Non-Parametric Methods: Nonparametric Local Regression

Laura Mayoral

IAE and BSE

Barcelona, Spring 2023

# 1. Introduction

- Goal: estimate the relationship between y and x without imposing a functional form.

$\rightarrow$ the same, in more technical words: nonparametric estimation of the conditional expectation.

- The conditional expectation of $Y$ conditional on $X$ (a univariate variable) at $x_0$:

$$E[Y|X = x_0] = m(x_0)$$

m(.) is not specified.

- We will start by considering that $X$ is a scalar variable (recall the curse of dimensionality)

- In this lecture we will develop nonparametric regression techniques

- These nonparametric methods are local averaging methods: estimates are obtained by cutting the data into ever smaller slices as $N \to \infty$ and estimating local behavior within each slice.

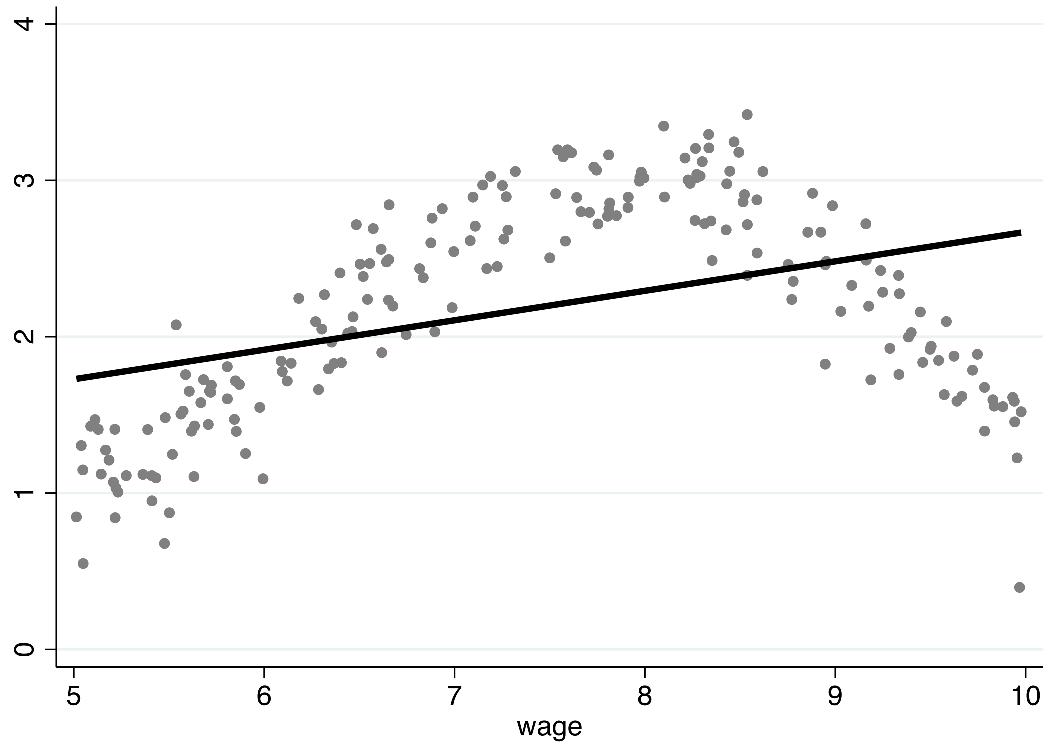
- In a nutshell: for each point $x_0$ those estimators are weighted (typically, kernel weights) local (=in a neighbor of $x_0$) averages of values of $y$

# An introductory example

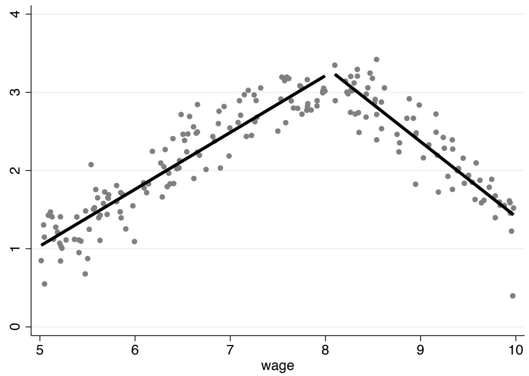
- Consider the relationship between hours worked per day and hourly wage (simulated data we created in handout1)
- We run an OLS regression and get a very positive relationship

```
reg y x

      Source |       SS           df       MS      Number of obs   =       200
-------------+----------------------------------   F(1, 198)       =     39.00
       Model |  15.6516595         1  15.6516595   Prob > F        =    0.0000
    Residual |  79.4533954       198  .401279775   R-squared       =    0.1646
-------------+----------------------------------   Adj R-squared   =    0.1604
       Total |  95.1050549       199  .477914849   Root MSE        =    .63347

------------------------------------------------------------------------------
           y | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
           x |   .1885303   .0301873     6.25   0.000     .1290004    .2480602
       _cons |   .7854619   .2283455     3.44   0.001     .3351607    1.235763
------------------------------------------------------------------------------
```

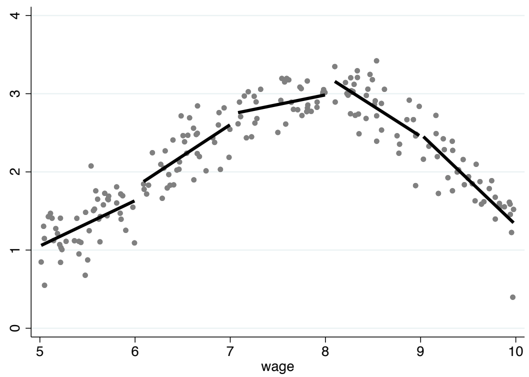
- However, plot the data: highly nonlinear relationship



- We could estimate two lines, one for the increasing part of the relationship and one for the decreasing one:




- Or we could even consider more regression lines, (i.e., a smaller "bandwidth")



- The previous methods will work if we know the breaking points (we're imposing them when running the OLS regressions).

- Nonparametric methods share a similar spirit: they are local averaging methods.

- No need to impose any breaking points as in this example!

- estimates are obtained by cutting the data into ever smaller slices as $N \to \infty$ and estimating local behavior within each slice.

- Parametric versus Nonparametric methods:

- Asymptotic properties are quite different

  - Lower convergence rates: because of local averages (less than less than N observations in estimating each slice)

  - In simplest cases still asymptotically normally distributed;

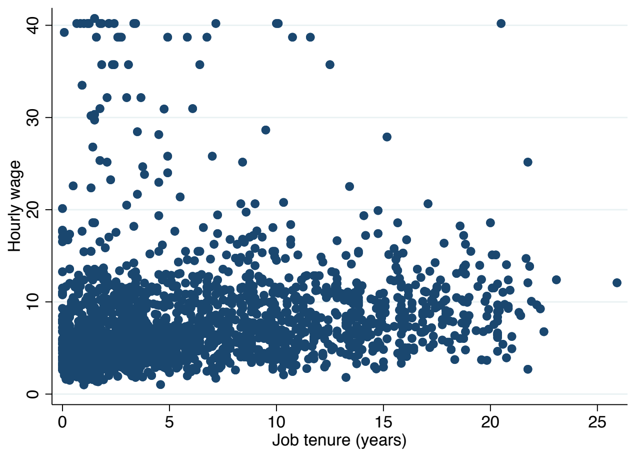
  - Due to lower convergence rates, biases appear

- Things become in general a bit “uglier” and properties are a bit ”less nice” than in parametric estimation, so be a bit patient!

## 1.2. Some simple visualization tools

- Before we begin with the complicated stuff, let's always look at the data first!

- Consider this example: The relation between tenure on the job and hourly wage.
- DATA (example STATA: sysuse nlsw88)
- Simplest visualization tool: scatterplot



- What can you say about the relationship between wage and tenure by looking at this graph?

- Not much!

- scatterplots are not very useful for large data sets

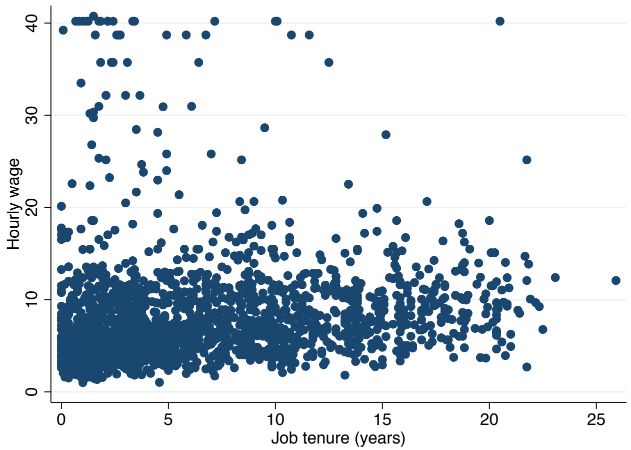
# A better way of plotting the data: binned scatter plots

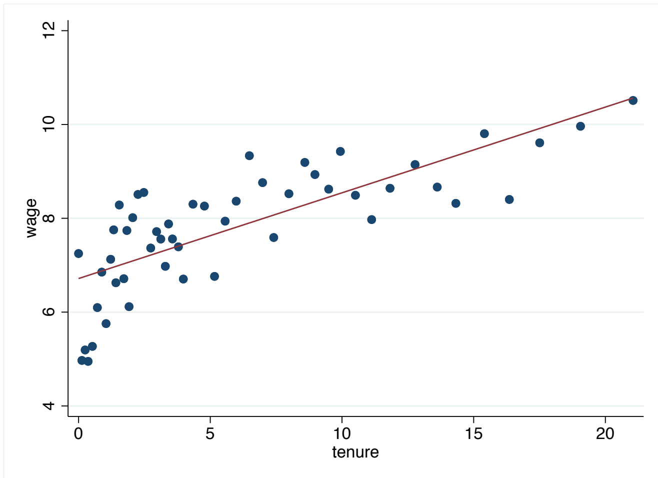
- If a lot of data points: scatter plots are not very useful (clouds of millions points! impossible to see anything)

- Binned scatters: very useful visualization tools, particularly for large datasets

- STATA: binscatter command

- Compare the scatter and the binned scatter plot (on same data)



- The second graph is much more informative that the first one!
- From the second graph, you can easily see that
  - There's a positive relationship between tenure and wage
  - This relationship seems to be pretty linear

- What's the magic? Binned scatter plots are visual, simple, non parametric estimators of the conditional expectation.

- How they work (from STATA help)

Binscatter command:

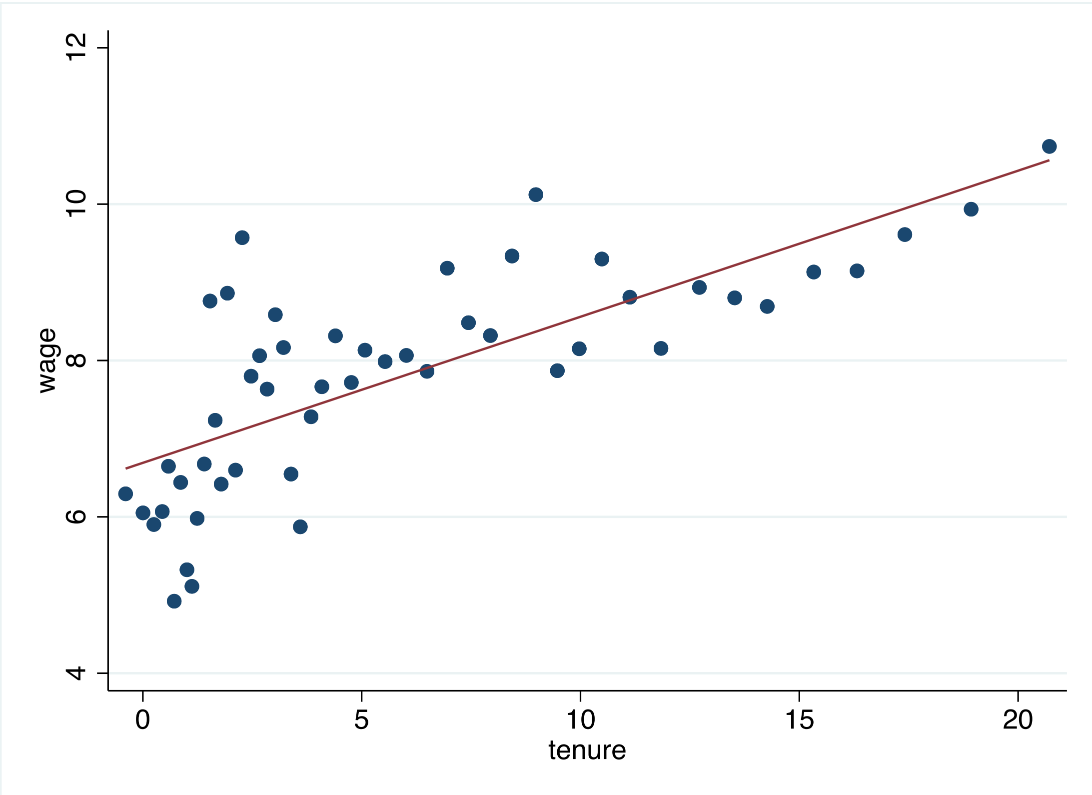
- groups the x-axis variable into equal-sized bins (number of bins to be determined by you, default 20)
- computes the mean of the x-axis and y-axis variables within each bin (median is also an option)
- then creates a scatterplot of these data points.
- The result is a non-parametric visualization of the conditional expectation function.

- Additional options of the binscatter command:

1. You can very easily control for other variables that you might think are relevant.

Control for age.

binscatter wage tenure, control(age) nq(50)



- Note: How does binscatter deal with control variables?

- Method inspired by a famous theorem in regression analysis: The Frisch-Waugh-Lowell Theorem

- Binscatter residualizes the x-variable and y-variables on the specified controls before binning and plotting.

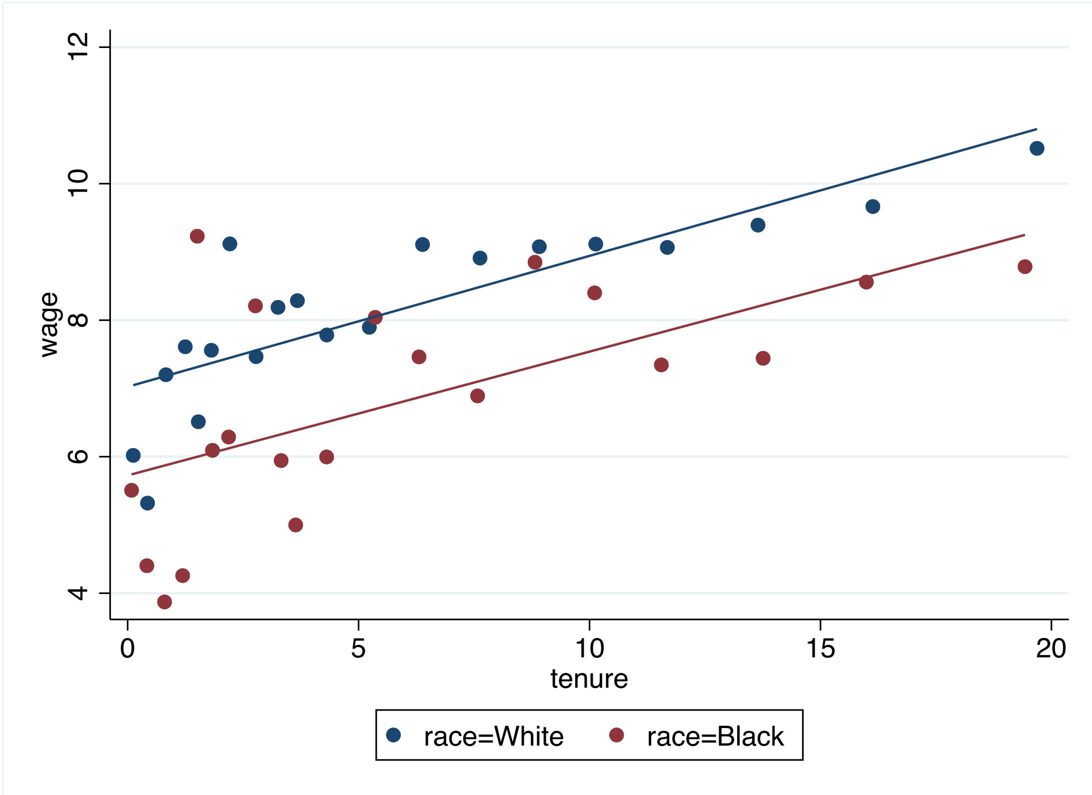
- That is,

  - regressing y/z, take residuals, add mean of y
  - regressing x/z, take residuals, add mean of x

2. You can plot the data for values of other variable.

For instance, by race

binscatter wage tenure, by(race) nq(50)



STATA code to generate this example:

- Load data in stata memory:

```
sysuse nlsw88

keep if inrange(age,35,44) & inrange(race,1,2)

keep if inrange(age,35,44) & inrange(race,1,2)

scatter wage tenure, graphregion(color(white)) lwidth(thick)

binscatter wage tenure, nq(50)

binscatter wage tenure, control(age) nq(50)

binscatter wage tenure,by(race) nq(50)
```

# Takeaway

- Always start by plotting your data

- Binned scatterplots are very useful tools, particularly when a lot of data points

  - Visual and quick estimator of the conditional expectation

  - Quite flexible stata command, allows to eliminate impact of other variables (linearly)

- But binscatter is not enough! (no inference, a bit too crude...)

# Overview of the handout

- The remaining of this handout: Different approaches to carry out nonparametric regression.

- Different methods: Kernel local (constant) regression; Local linear/polynomial regression; Lowess, . . .

- Intuition is simple, technical stuff becomes complicated

- We will look at

1) Intuition;

2) implementation

3) differences across the methods; etc

4) stata tips (more on this in the TA session)

# Roadmap of this handout

1. Introduction: Nonparametric Local Regression;

   1.2. Some simple visualization tools

2. Local Weighted Averages

3. Kernel Local Regression: implementation, properties

4. Local Linear Regression

5. K-Nearest Neighbor

6. Lowess

# 2. A bif of intuition: Local Weighted Averages

- Model:

$$y_i = m(x_i) + \epsilon_i, \quad i = 1, \ldots, N, \quad \epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon). \tag{1}$$

and $E(\epsilon|x) = 0$.

- Under these assumptions, the conditional expectation is

$$m(x_0) = E(y|x = x_0). \tag{2}$$

- Problem:

m(.) unspecified $\rightarrow$ use nonparametric methods to estimate it at a point $x_0$

# A bit of intuition about local average estimators

- Suppose that at $x_0$, there are multiple observations on $y$, say $N_0$ observations.

- A simple estimator for $m(x_0)$ is the sample average of these $N_0$ values of $y$.

$$\hat{m}(x_0) = \sum_{i=1}^{N_0} w_i y_i$$

where $w_i = 1/N_0$ if $x = x_0$ and 0 otherwise.

- Notice that (for fixed $x_0$):

$$\bar{m}(x_0) \sim \left( m(x_0), \frac{\sigma^2}{N_0} \right), \tag{3}$$

- Why? it is the average of $N_0$ observations that are i.i.d with mean $m(x_0)$ and variance $\sigma_\epsilon^2$.

- The estimator $\bar{m}(x_0)$ is unbiased but not consistent (in general)

  - Why? Consistency requires $N_0 \to \infty$ as $N \to \infty$, so that $V[\bar{m}(x_0)] \to 0$.

  - But $N_0$ can be really small, particularly for continuous variables! (most likely, just one observation of y)

Then:

- The Problem of this approach: not enough observations to average ($N_0$ can be too small, it can even be 1 for continuous variables even with a huge sample!)

- A Solution: consider averages of $y$ when $x$ is close to $x_0$, (in addition to when $x$ exactly equals $x_0$).

## Local weighted average estimator:

- a weighted average of the dependent variable in a neighborhood of $x_0$.

$$\widehat{m(x_0)} = \sum_{i=1}^{N} w(x_i, x_0, h) y_i$$

where the weights $w(x_i, x_0, h)$ sum to 1 and vary with :

- the sample values of the regressors, $x_i$
- the evaluation point $x_0$
- the value of $h$, i.e., the length of the window around $x_0$

- h: bandwidth parameter. Smaller values of h $\rightarrow$ smaller window $\rightarrow$ more weight being placed on those observations with $x_i$ close to $x_0$.
- 2h: window width

- The most common weight functions are:
- 1. Kernel weights
- 2. Lowess
- 3. k-nearest neighbors

- Modus operandi: compute $\widehat{m(x_0)}$ at a variety of points of $x_0$ to obtain a regression curve.

# 3. Kernel regression: NW estimator

- Recall the Model:

$$y_i = m(x_i) + \epsilon_i, \quad i = 1, \ldots, N, \tag{4}$$

$$E(\epsilon|x) = 0,$$
$$E(\epsilon^2|x) = \sigma^2(x)$$

.

- Recall the Goal: Estimate of $m(x_0)$,

$$m(x_0) = E(y|x = x_0). \tag{5}$$

- Let's now analyze the case where we use Kernel weights

- Kernel regression is a weighted average estimator using kernel weights.

- Consider again the local weighted average estimator, where we compute the average of the y's in an interval of length $2h$ around $x_0$

$$\widehat{m(x_0)} = \frac{\sum_{i=1}^{N} 1(|\frac{x_i - x_0}{h}| < 1) y_i}{\sum_{i=1}^{N} 1(|\frac{x_i - x_0}{h}| < 1)}$$

- The numerator: sums the y's in the interval $(x_0 \pm h)$

- The denominator: gives the total number of y's that have been summed in the numerator

- Thus: the previous expression is an average of the y's with equal weights (weights are relative frequency of $y$ in the window)

- Consider instead Kernel weights

- Why?

- non-constant weights

- give more weight to observations close to $x_0$

- Kernel Regression Estimator

$$\widehat{m(x_0)} = \frac{\sum_{i=1}^{N} K(\frac{x_i - x_0}{h}) y_i}{\sum_{i=1}^{N} K(\frac{x_i - x_0}{h})}$$

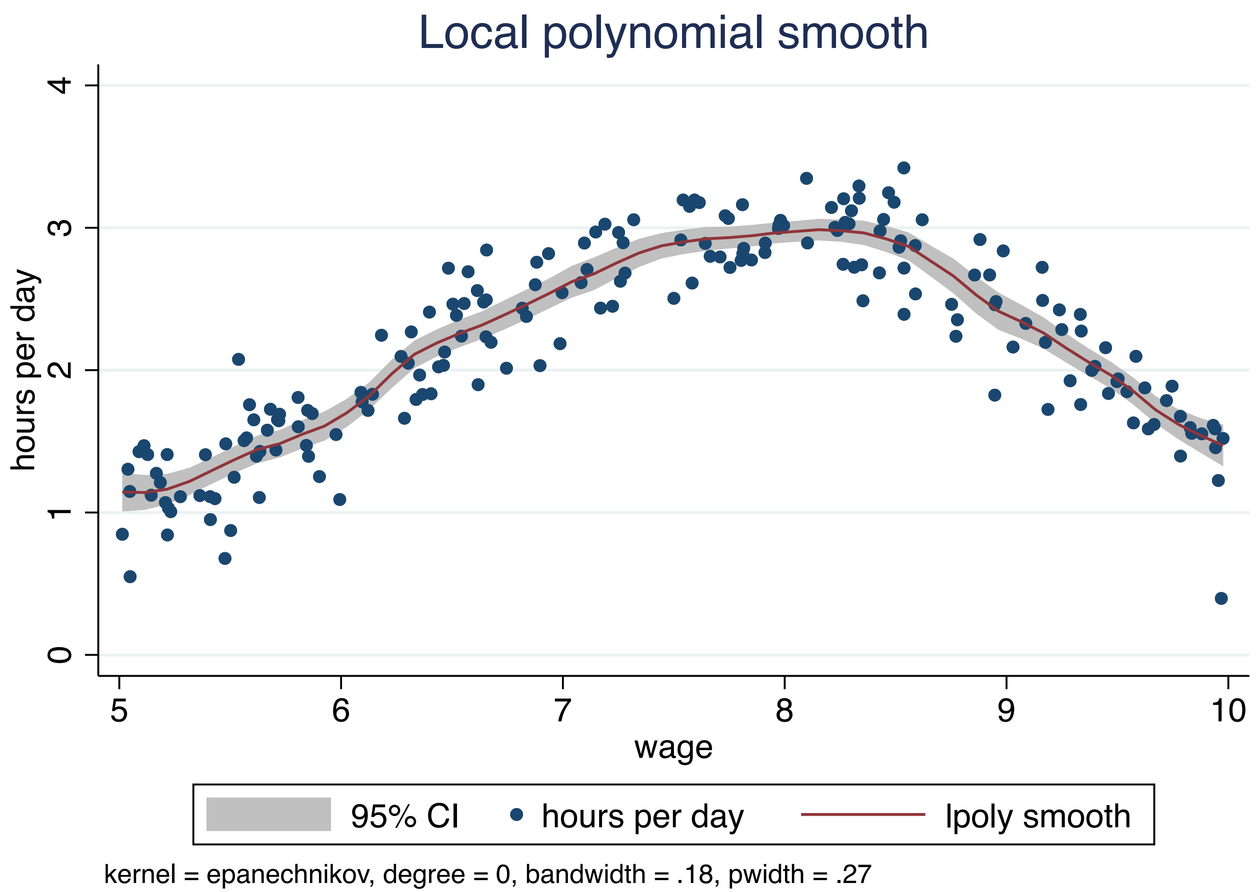
(also called Nadaraya-Watson estimator)

- similar Kernels as before: Gaussian, Epanechnikov, etc.

# Example: Nadaraya-Watson estimator for the hours worked /wage problem

(stata defaults for h, kernel... –we'll learn about them)

lpoly y x , ci msize(small) graphregion(color(white))



# Implementation of the NW estimator

## 1. Kernel choice

- Kernel choice: $MISE(h^*)$ is minimized by the Epanichnikov Kernel (as before)

- but small differences across kernels for optimal $h^*$

- Choice of bandwidth is much more important than choice of kernel

# Implementation of the NW estimator, II

## 2. Bandwidth choice

- Optimal bandwidth: recall the tradeoff between bias&variance in the choice of h.

- Optimal bandwidth: trades off bias (minimized with small bandwidth) and variance (minimized with large bandwidth)

- Variance=$O_p(Nh)$; bias=$O(h^2)$

- Theory just says that the optimal bandwidth (=the one that minimizes MISE) for kernel regression is $O(N^{-0.2})$ (but this is useless for choosing $h$ in applications).

- In practice: plug-in estimator of the optimal $h$ using MISE(h) is complicated now (estimation of the plug-in estimation requires estimation of $m''(x)$, second derivative of conditional expectation which is difficult to estimate).

- Alternative: Cross-validation, computationally intensive, but easier to implement

# Choosing the bandwidth: Cross-validation

- Cross-validation is a popular techniques for many prediction problems
- Cross-validation, in general:

- Construct prediction models that perform well out of sample
- Simple idea:
  - we split the data in two sets: training set and validation set
  - Use the data in the training set to construct the estimator.
  - Using this estimator, predict the "out of sample" observations, i.e., the obs. in the "validation set", calculate the error.
  - Choose the estimator with best out of sample performance

Why leaving some observations out?

- Avoid overfitting:

  - an estimator that is very good for the in-sample data but can perform badly for non-seen observations

  - why is that? because in a dataset there's always noise. If we perfectly fit that data, we fit both the "signal" (what really matters in the data) AND the noise, something that is pure random variation.

  - Since the noise changes in every realization of the data, a model that fits very well a dataset can perform badly out of sample

- Cross validation, in particular:

- Goal: use cross-validation to choose a value of $h$ that yields a good estimate $m(x)$

- Idea

  - For each observation $i$, compute an estimator $m_i$ using cross-validation i.e., using a "training sample" to compute the estimator

  - ...and a validation sample only used to compute out of sample prediction error

  - Then, choose h that yields smallest MSE.

- How it works (a bit simplified):

- 1. For each $i$, define the training sample as all the observations except obs. $i$; validation sample: observation $i$

- 2. The estimator leaving $i$ out is given by

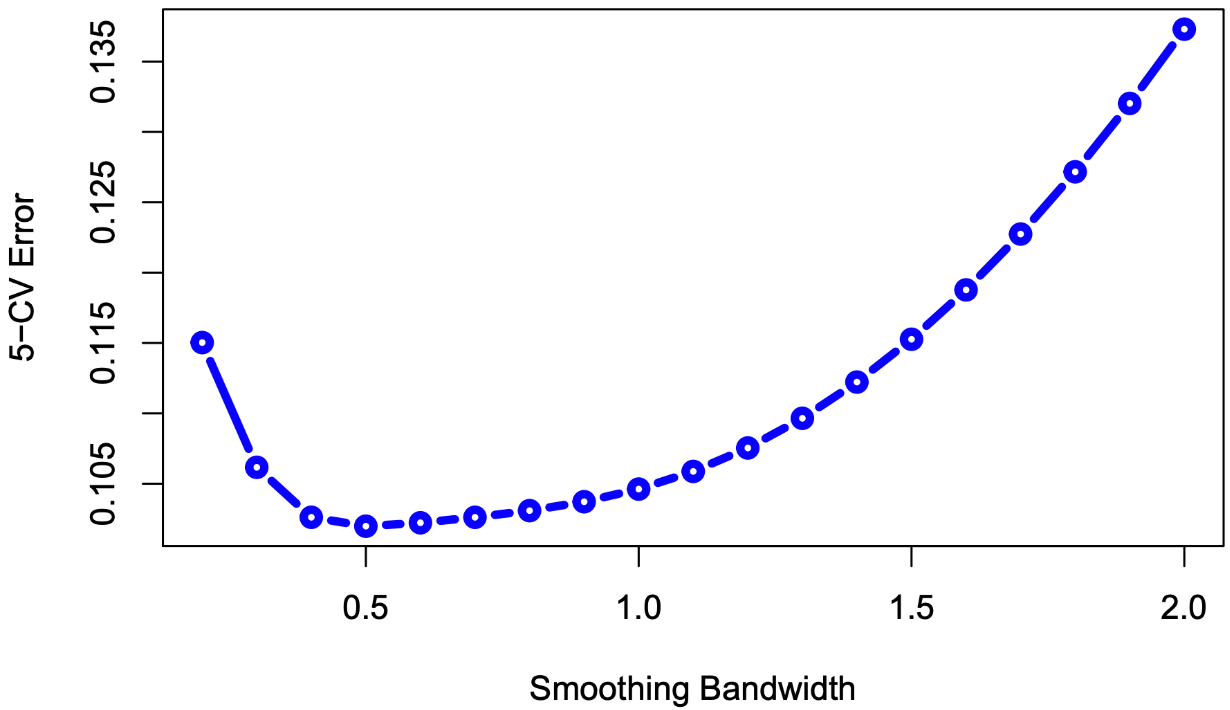
$$\hat{m}_{-i}(h, x_i) = \sum_{j \neq i} w_{j,h} y_j \Big/ \sum_{j \neq i} w_{j,h}$$

- 3. Compute CV(h) (very similar to the MSE(h))

$$CV(h) = \sum_{i=1}^{n} \left(y_i - \hat{m}_{-i}(x_i)\right)^2 \pi(x_i), \tag{6}$$

- $\pi(x_i)$: weights introduced to potentially downweight the end points, to prevent those points to receive too much attention (local weighted estimates can be quite highly biased at the end points)

- 4. $h^*_{cv}$ is chosen as the value that minimizes the CV(h)
- 5. In practice CV(h) is computed over a range of values of h. Choose the value of h that makes it smallest.
- Properties of $\hat{h}_{cv}$: converges to $h^*$ (optimal h), but slowly ($\approx$ low convergence rate)



## Takeaway

- In Kernel regression, cross-validation tends to perform better than the plug-in estimator

- Logic of Cross-validation: choose the $h$ that minimizes the (out of sample) mean prediction error

- Why leaving one observation out at a time?

- nonparametric methods are very flexible, and if we consider the whole sample, we can get an almost "perfect fit"

- $\Rightarrow$ Overfitting!

# Statistical Properties of Kernel regression estimators

## 1. The Kernel regression estimator is consistent

- $\widehat{m_0}$ is consistent if some conditions on $h$ and $Nh$ hold

- Recall: these conditions are needed for developing the theory; not informative to choose the value of h in practice

- The estimator is consistent provided:

  - $h \to 0$: i.e., substantial weight is given only to $x_i$ very close to $x_0$.

AND

$Nh \to \infty$: i.e., there's "many" $x_i$ close to $x_0$ as $n \to \infty$, so that many observations are used in forming the weighted average.

## 2. The Kernel regression estimator is biased in finite samples

- It can be shown that

$$\widehat{m(x_0)} = m(x_0) + O(h^2)$$

- Asymptotically, the bias tends to zero under the assumptions above (i.e. h tends to zero)

- However, the bias can be substantial in finite samples

- Particularly, at the end points (where few observations exist)

- When considering confidence intervals, the estimate is centered in the true value of $m$ plus the bias!

## 3. The Kernel regression estimator is asymptotically normal

- Rate of converge: $\sqrt{(}Nh)$: smaller than the usual $\sqrt{(}N)$

- Asympotic distribution (notice the bias!)

$$\sqrt{Nh}(\hat{m}(x_0) - m(x_0) - b(x_0)) \rightarrow N\left(0, \frac{\sigma_\epsilon^2}{f(x_0)} \int K(z)^2 dz\right) \tag{7}$$

# Constructing Confidence Intervals

- Estimates of $m(x_0)$ typically are provided with CI

- How can we compute them?

1. Use the asymptotic distribution above ignoring the bias. Then:

$$m(x_0) \in \hat{m}(x_0) \pm 1.96 \sqrt{\frac{1}{Nh} \frac{\hat{\sigma}_\epsilon^2}{\hat{f}(x_0)} \int K(z)^2 dz}$$

- But two problems

Problem 1 Convergence to the normal distribution is slow (recall the lower convergence rates)

Problem 2 Forgetting the bias means that the CI are not centered correctly!

- Solutions.

Problem 1. Don't use the asympotic distribution, instead use bootstrap (i.e., a method that approximates the finite sample distribution)

Problem 2. Reduce the bias: a) Undersmoothing and/or b) using higher order Kernels (Fourth-order, Gaussian Fourth-order quartic); c) Use alternative methods that are less biased: Local polynomial regression, Lowess . . . (smaller bias)

- Notice that it's possible to combine the solutions to problems 1 and 2, as they tackle different problems.

For instance, you can use bootstrad AND undersmoothing

# Example

- DATA: PSID Individual Level Final Release 1993 data, (www.isr.umich then choose Data Center )

- Relation between years of completed education and (log of) wages
- Females in their 30's
- Data from Cameron and Trivedi

- OLS regression:
- highly significant role of education;

```
regress lnhwage educatn

      Source |       SS           df       MS      Number of obs   =       177
-------------+----------------------------------   F(1, 175)       =     25.19
       Model |   15.189945         1   15.189945   Prob > F        =    0.0000
    Residual |  105.519895       175  .602970827   R-squared       =    0.1258
-------------+----------------------------------   Adj R-squared   =    0.1208
       Total |   120.70984       176  .685851362   Root MSE        =    .77651

------------------------------------------------------------------------------
     lnhwage | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
     educatn |   .1033945      .0206     5.02   0.000     .0627381     .144051
       _cons |   .8966776   .2657917     3.37   0.001     .3721077    1.421247
------------------------------------------------------------------------------
```

- interpretation : marginal effect

- OLS regression:
- highly significant role of education;

```
regress lnhwage educatn

      Source |       SS           df       MS      Number of obs   =       177
-------------+----------------------------------   F(1, 175)       =     25.19
       Model |   15.189945         1   15.189945   Prob > F        =    0.0000
    Residual |  105.519895       175  .602970827   R-squared       =    0.1258
-------------+----------------------------------   Adj R-squared   =    0.1208
       Total |   120.70984       176  .685851362   Root MSE        =    .77651

------------------------------------------------------------------------------
     lnhwage | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
     educatn |   .1033945      .0206     5.02   0.000     .0627381     .144051
       _cons |   .8966776   .2657917     3.37   0.001     .3721077    1.421247
------------------------------------------------------------------------------
```

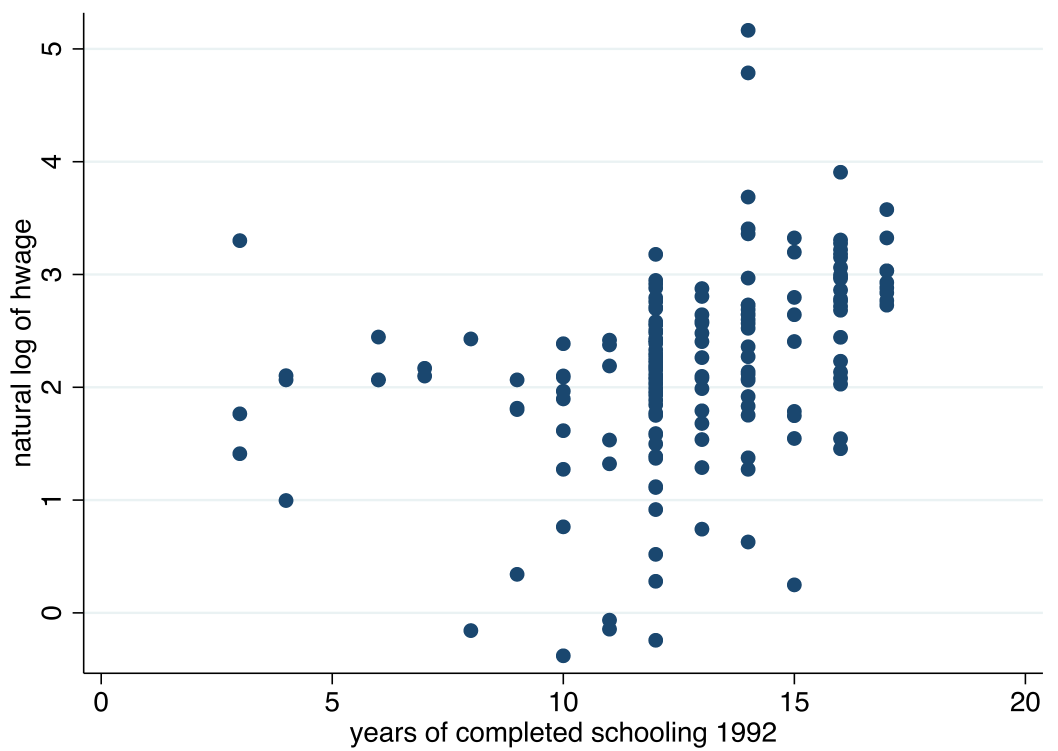
- interpretation : marginal effect

an increase in one year of education increases by 10% hourly wage.

# But...is the linearity assumption reasonable?

Let's plot the data (scatter plot)

twoway scatter lnhwage educatn, graphregion(color(white))



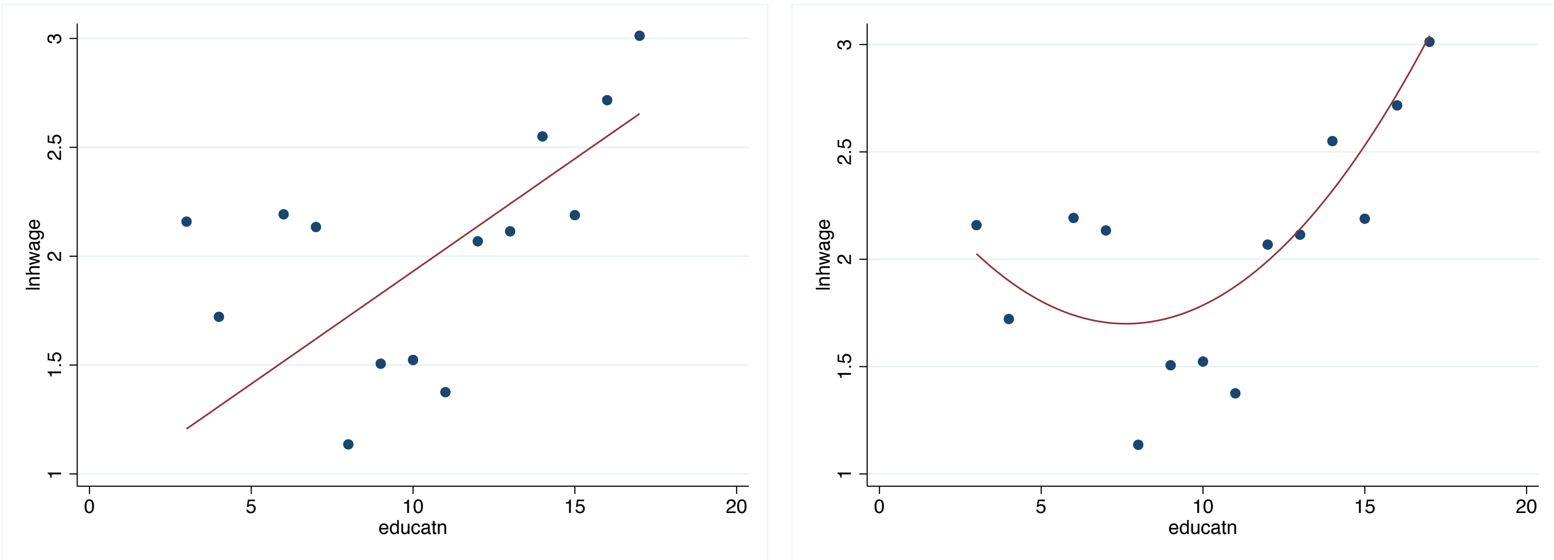
# Binned scatter plot

- STATA: binscatter command

binscatter lnhwage educatn, nq(20)

binscatter lnhwage educatn, nq(20) line(qfit)

(first graph imposes a linear fit on the data, second is more flexible, allows for a quadratic one)



# Nonparametric regression in STATA

## The lpoly and npregress commands

- STATA has several commands to do nonparametric regression: lpoly, npregress (the latter has more options)

- lpoly: Kernel-weighted local or polynomial smoothing
- Less options than npregress
- Very easy to use

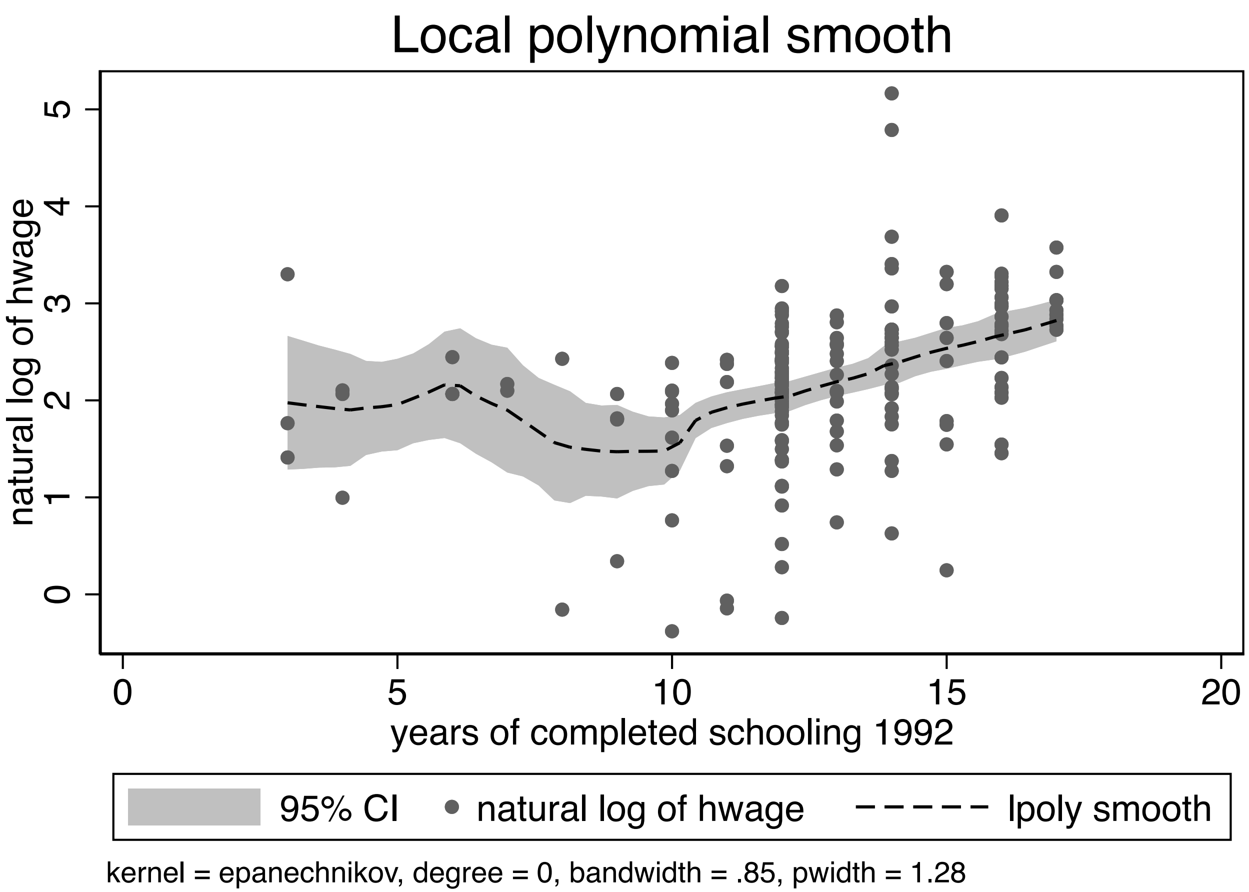
- npregress kernel:
- From Stata 15 onwards: a new command, npregress
- Determines bandwidth by cross-validation whereas lpoly uses plug-in value
- Evaluates at each $x_i$ value (whereas lpoly default is to evaluate at 50 equally spaced values)
- For local linear, computes partial effects.
- Can use margins and marginsplot for plots and average partial effects.

- Can deal with more than one regressor.
- we'll see an example in a few slides

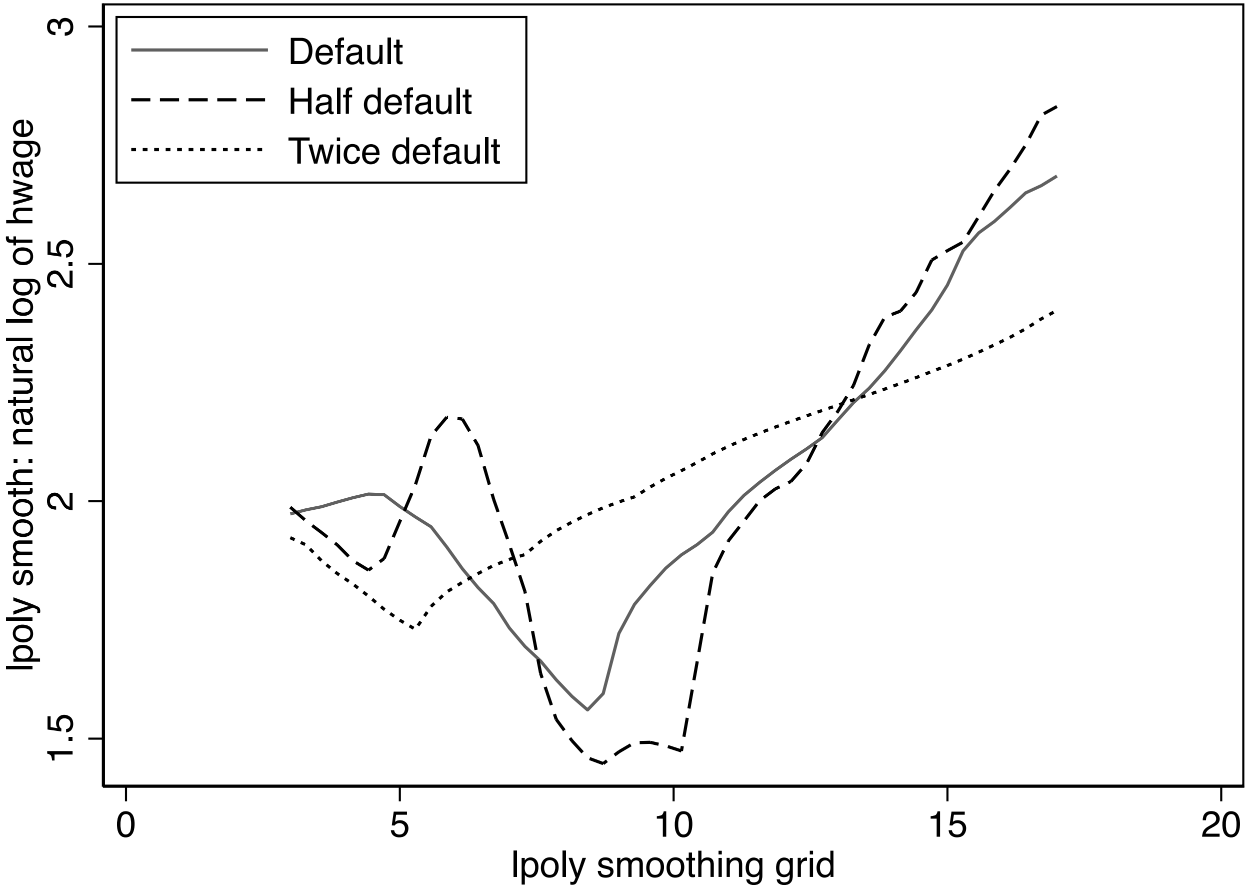
# Example

lpoly lnhwage educatn, ci

(STATA default values, default is degree 0 –constant–; plug in estimator)



- Try different values for the bandwidth



# Takeaway

- First nonparametric regression method: Kernel local regression

- In a nutshell: local averages of the dependent variable, $y$

- Choice of bandwidth is key

- Use cross-validation to select $h$

- Choice of kernel is less important, optimal kernel: Epanechnikov

- STATA commands: npregress, lpoly

- Asymptotic properties: consistent, asymptotically normal

- Lower convergence rates (biases, non centered CI . . . )

# 4. Other methods: Local Linear Regression

- The Nadaraya–Watson estimator can be seen as a particular case of a wider class of nonparametric estimators, the so-called local polynomial estimators.

- The Nadaraya–Watson estimator is a local constant estimator because it assumes that $m(x)$ equals a constant in the local neighborhood of $x_0$.

- Now: let $m(x)$ be linear in the neighborhood of $x_0$,

$$m(x) = a_0 + b_0(x - x_0) \text{ in the neighborhood of } x_0$$

.

# Implementation of this idea

1) Notice that the kernel regression estimator (previous estimator) $m(x_0)$ can be obtained as

$$\widehat{m(x_0)} = argmin_{m_0} \sum_i W(\frac{x_i - x_0}{h})(y_i - m_0)^2$$

where the weights are the NW weights:

$$W(x_i) = K(x_i - x_0) / \sum_{j=1}^{N} K(x_j - x_0)$$

- Why? remember that $m(x_0)$ is a constant and $e_i = y_i - m_0$. Then, this is similar as weighted least squares, $e_i = y_i - m_0$.

2) Consider now $m_0 = a_0 + a_1(x_i - x_0)$. Obtain the local linear estimator as:

$$\widehat{m(x_0)} = argmin_{a_0,a_1} \sum_i W(\frac{x_i - x_0}{h})(y_i - a_0 - a_1(x_i - x_0))^2$$

Then, the estimate of $m$ is a neighborhood of $x_0$ is given by

$$\hat{m}(x) = \hat{a}_0 + \hat{a}_1(x - x_0)$$

- Same idea: this is (local) weighted least squares regression, where the weights are kernel weights

- Interpretation:

- The constant $a_0$ is the conditional mean at $x_0$.
- The slope parameter, $a_1$: is the derivative of the mean function with respect to $x$.

3) More generally, we can consider a local polynomial estimator of degree p

$$argmin_{a_0,a_1} \sum_i W(\frac{x_i - x_0}{h})(y_i - a_0 - a_1(x_i - x_0) \cdots - a_p(x_i - x_0)^p)^2$$

## Some advantages over NW

- Higher accuracy: Local linear regression estimators use a more flexible model that allows for a more accurate fit to the data, especially in regions where the data may be changing rapidly. Better behavior at end points (always problematic because of low density of data points).

- Easy computation of derivatives: (very useful for interpreting results)

- Cons: A bit more costly computationally than NW

## Example

- Consider again the education/wage example: we will estimate a local linear regression

- STATA: Can be estimated using lpoly or npregress

lpoly lnhwage educatn,degree(1).

Or npregress kernel lnhwage educatn –several options available!–

- Let's look at the latter

- npregress command - default is local linear

- The output reports averages of the mean function and the effects of the mean function.

- An average effect may be either 1) an average marginal effect, for continuous covariates or 2) the mean of contrasts for discrete covariates.

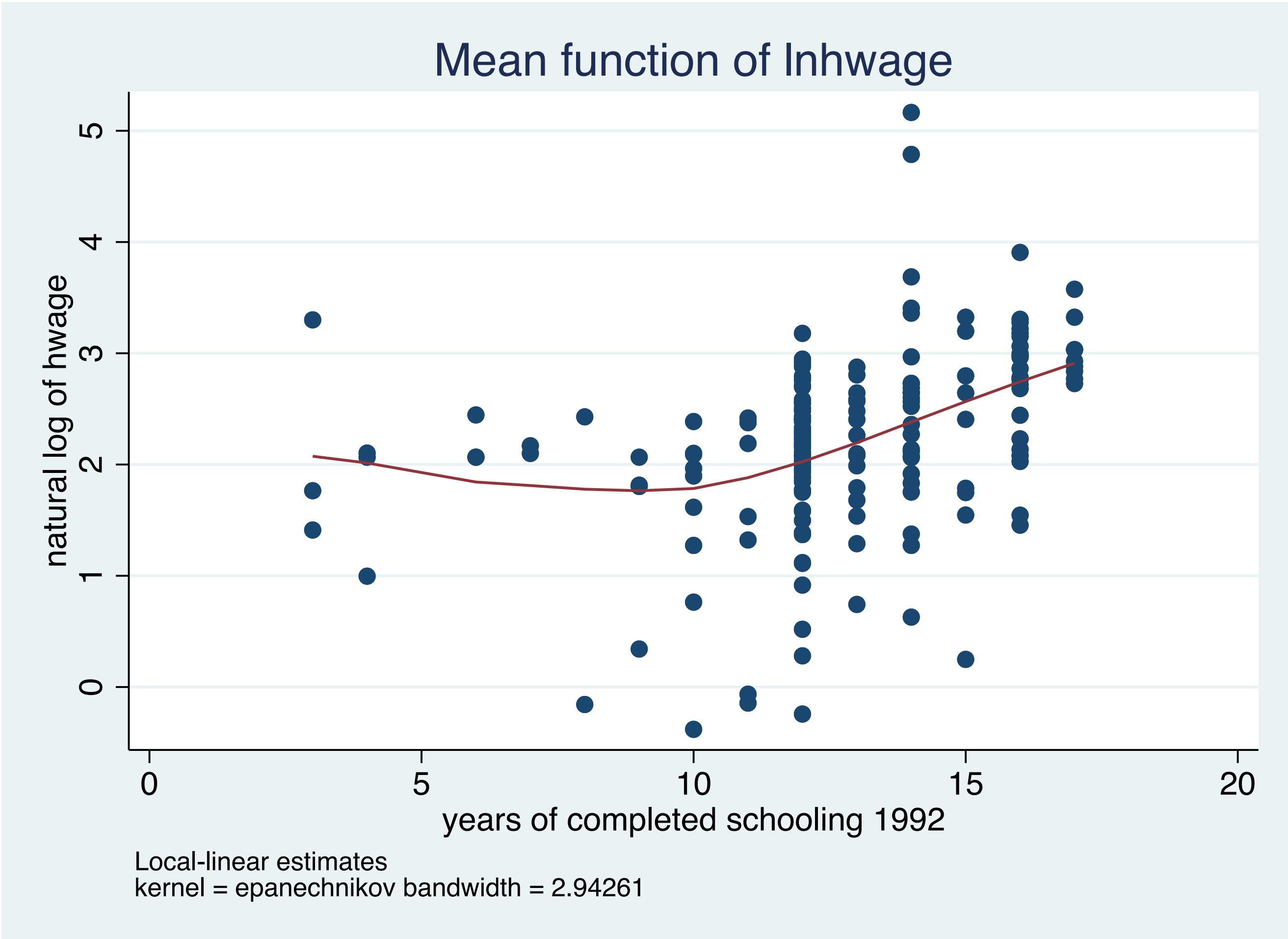
npregress kernel lnhwage educatn

- **npregress** reports averages $\widehat{\alpha} = \frac{1}{N}\sum_{i=1}^{N} \widehat{\alpha(x_i)}$ and $\widehat{\beta} = \frac{1}{N}\sum_{i=1}^{N} \widehat{\beta(x_i)}$

```
Bandwidth
─────────────────────────────────────
              │     Mean      Effect
──────────────┼──────────────────────
Mean          │
      educatn │  2.94261    4.004823
─────────────────────────────────────

Local-linear regression                    Number of obs      =        177
Kernel   : epanechnikov                    E(Kernel obs)      =        177
Bandwidth: cross validation                R-squared          =     0.1943
──────────────────────────────────────────────────────────────────────────
      lnhwage │   Estimate
──────────────┼───────────────────────────────────────────────────────────
Mean          │
      lnhwage │   2.223502
──────────────┼───────────────────────────────────────────────────────────
Effect        │
      educatn │   .1492393
──────────────────────────────────────────────────────────────────────────
Note: Effect estimates are averages of derivatives.
Note: You may compute standard errors using vce(bootstrap) or reps().
```

- Versus OLS $\widehat{\alpha} = 0.897$ and $\widehat{\beta} = 0.10$

npgraph:



Mean function of lnhwage
natural log of hwage
0
1
2
3
4
5
0
5
10
15
20
years of completed schooling 1992
Local-linear estimates
kernel = epanechnikov bandwidth = 2.94261

Obtain bootstrap standard errors and confidence intervals for these values

npregress kernel lnhwage educatn, vce(bootstrap, seed(10101) reps(50))

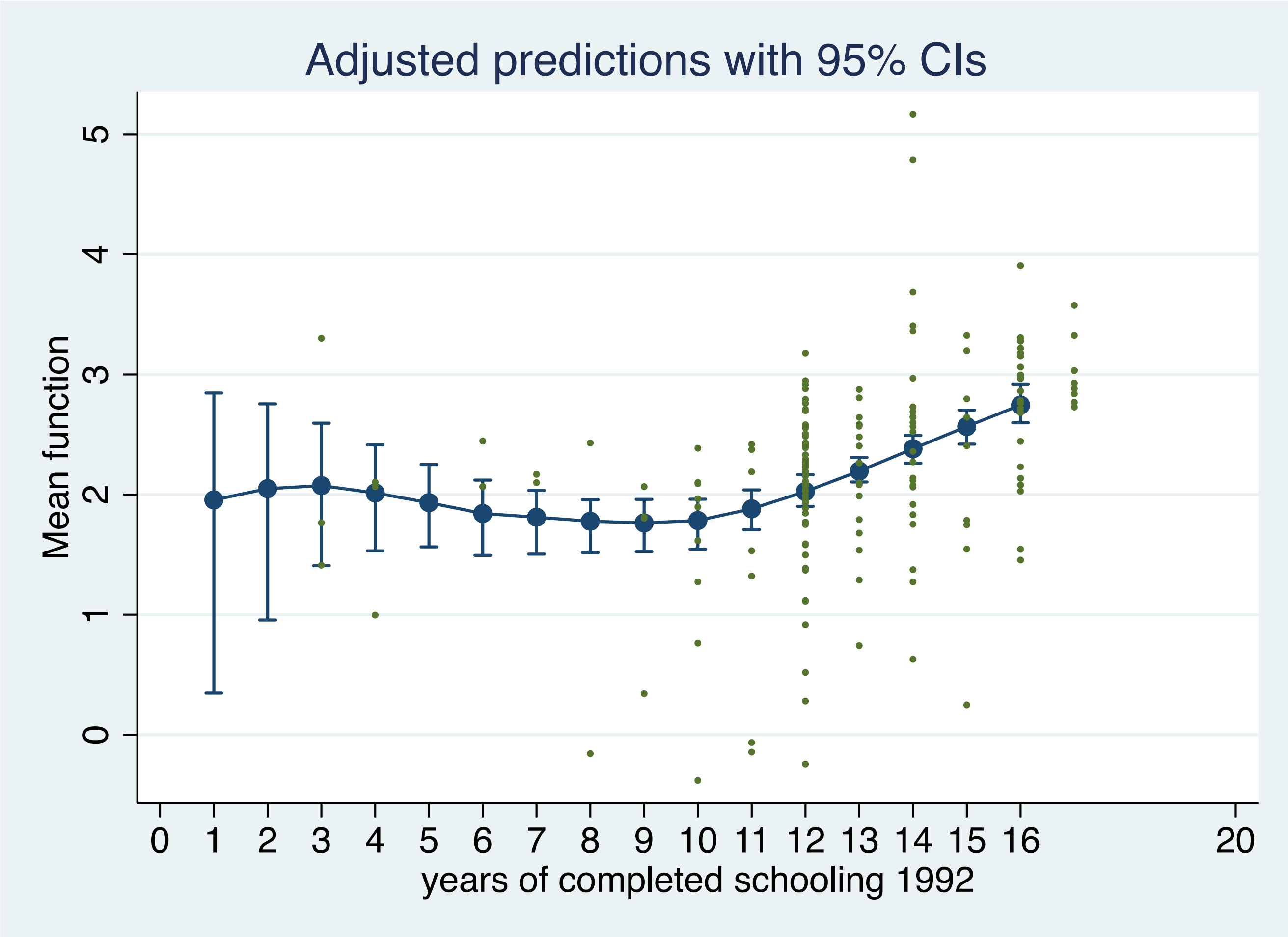
- Get bootstrap standard errors

```
. * npregress with bootstrap standard errors
. npregress kernel lnhwage educatn, vce(bootstrap, seed(10101) reps(50))
(running npregress on estimation sample)

Bootstrap replications (50)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50

Bandwidth
------------------------------------
             |      Mean     Effect
-------------+----------------------
Mean         |
     educatn |   2.94261   4.004823
------------------------------------

Local-linear regression                    Number of obs      =        177
Kernel   : epanechnikov                    E(Kernel obs)      =        177
Bandwidth: cross validation                R-squared          =     0.1943
------------------------------------------------------------------------------
             |   Observed   Bootstrap                         Percentile
     lnhwage |   Estimate   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
Mean         |
     lnhwage |   2.223502   .0635099    35.01   0.000     2.121183      2.3635
-------------+----------------------------------------------------------------
Effect       |
     educatn |   .1492393   .0242175     6.16   0.000      .114171    .1941928
------------------------------------------------------------------------------
Note: Effect estimates are averages of derivatives.
```

- Versus OLS se$(\widehat{\alpha}) = 0.302$ and se$(\widehat{\beta}) = 0.023$.

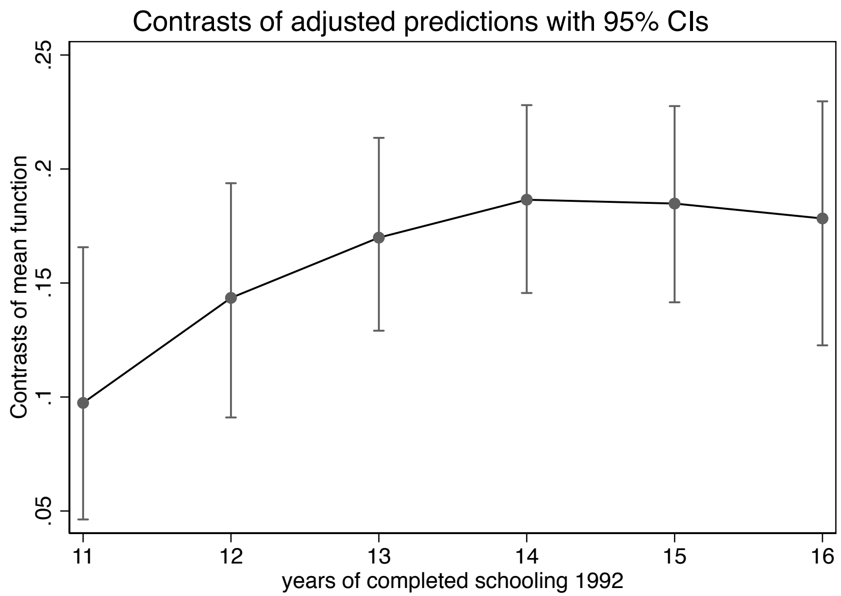
Plot the graph

margins, at(educatn = (1(1)16)) vce(bootstrap, seed(10101) reps(50))
marginsplot, legend(off) scale(1.1) /// addplot(scatter lnhwage educatn if lnhwage<50000, msize(tiny))



- Partial effects of changing education

```
margins, at(educatn = (10(1)16)) contrast(atcontrast(ar)) ///
vce(bootstrap, seed(10101) reps(50))
marginsplot, legend(off)
```



- Stata code

```
npregress kernel lnhwage educatn

npregress kernel lnhwage educatn, vce(bootstrap, seed(10101) reps(50))

margins, at(educatn = (10(1)16)) vce(bootstrap, seed(10101) reps(50))

marginsplot, legend(off) scale(1.1) /// addplot(scatter lnhwage
educatn if lnhwage¡50000, msize(tiny))

graph export nonparametricfig11.wmf, replace

margins, at(educatn = (10(1)16)) contrast(atcontrast(ar)) ///

vce(bootstrap, seed(10101) reps(50))

marginsplot, legend(off)

graph export nonparametricfig13.wmf, replace
```

# 5. Nearest Neighbor Estimator

- Simple idea: The $k$-nearest neighbor estimator is the weighted average of the $y$ values for the $k$ observations of $x_i$ closest to $x_0$.

- Define $N_k(x_0)$: the set of $k$ observations of $x_i$ closest to $x_0$. Then:

$$m_{K-NN}(x_0) = \frac{1}{k} \sum_{i}^{N} 1(x_i \in N_k(x_0)) y_i$$

- This estimator is
  - a kernel estimator with uniform weights
  - except that the bandwidth is variable.

- Here the bandwidth $h_0$ at $x_0$ equals the distance between $x_0$ and the furthest of the $k$ nearest neighbors, and more formally $h_0 = k/(2Nf(x_0))$.

- Pros: a simple rule for variable bandwidth selection.

- It is computationally faster to use a symmetrized version that uses the k/2 nearest neighbors to the left and a similar number to the right

# 6. Lowess

- Lowess: locally weighted scatterplot smoothing estimator

- A variant of local polynomial estimation (kernel)

- Computational Differences:

  - uses a variable bandwidth $h_{0,k}$ determined by the distance from $x_0$ to its kth nearest neighbor;

  - tricubic kernel

  - Robust against outliers: downweights observations with large residuals $e_i = y_i - m(x_i)$, which requires passing through the data N times.

Lowess has some advantages with respect to local lineal regression:

- More robust against outliers

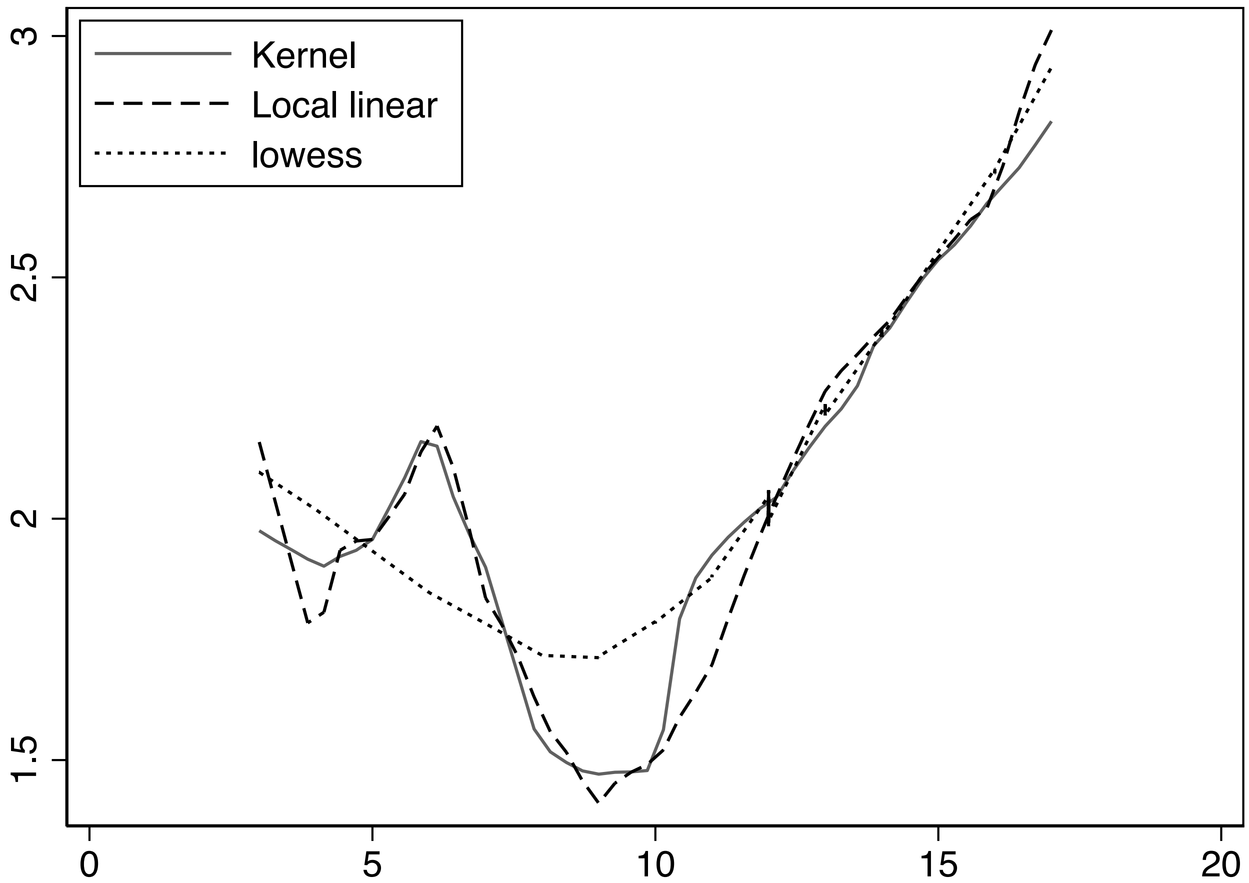
- But computationally more expensive

- See Fan and Gijbels (1996, p. 24). for additional details Lowess is attractive compared to kernel regression as it uses a variable

# Example

Comparison of local constant, local linear and lowess: wage and years of education

To compute lowess: (lowess lnhwage educ, clstyle(p3)), scale(1.1) ///



# Multivariate Kernel Regression:

- Conceptually, multivariate kernel regression is identical to univariate one

$$\hat{m}(x_0) = \sum_{i=1}^{N} W(x_i, x_0, h) y_i$$

where $x$ is a $k \times 1$ vector, $W(x_i, x_0, h) = K((x_i - x_0)/h) / \sum_i K((x_i - x_0)/h)$ and K(.) is a multivariate kernel

- Often, the multivariate kernel is just the product of univariate kernels

- If this is the case, divide by standard deviation so that all variables have similar scale

- Use cross validation to choose a common bandwidth $h^*$

- Important: convergence rates decreases (curse of dimensionality)
- Before: $\sqrt{Nh}$,
- Now: $\sqrt{Nh^k}$, where k is the number of covariates

# Takeaway

- So far: Kernel-based methods to visualize/estimate conditional expectation in a flexible way

- Methods based on local averages of the dependent variable

- Several methods: local Kernel, local polynomial, Lowess, nearest neighbor . . .

- Methods differ in bandwidth used, weights used, etc.

- Not huge differences, but Lowess and local polynomial behave better at end points.

- These methods can handle multivariate regression, but rates of convergence decrease, so performance deteriorates as the number of regressors increases.