University of California, Santa Cruz
Department of Economics
ECON 294A (Fall 2014) - Stata Lab
Instructor: Manuel Barron[1]

Econometric Tools 1: Non-Parametric Methods

1 Introduction

This lecture introduces some of the most basic tools for non-parametric estimation in Stata. Non-parametric econometrics is a huge field, and although the essential ideas are pretty intuitive, the concepts get complicated fairly quickly. This lecture is meant to give you some background knowledge of non-parametric methods in econometrics. If you are interested in using non-parametric methods more in depth, there are many textbooks at different levels of sophistication. For instance "Non-parametric Econometrics" by Pagan and Ullah is fairly accessible, but if you would like more advanced treatment (one year PhD level course) you may want to use Li and Racine's "Non-parametric Econometrics: Theory and Practice" textbook, which is the standard non-parametric econometrics textbook in graduate programs.

In parametric methods we need to make assumptions about the distribution of the disturbance term (for instance, normality) or about the shape of the relation between the variables under analysis (for instance, linearity). The main advantage of non-parametric methods is that they require making none of these assumptions.

The most basic non-parametric methods provide appealing ways to analyze data, like plotting histograms or densities. These methods also allow to plot bivariate relationships (relations between two variables). Since the results of non-parametric estimation are typically presented as graphs, it becomes essential to produce nicely formatted graphs, so in this lecture we'll get a bit deeper into graph options, choosing line width, colors, pattern, etc.

Non-parametric methods are very useful to study relations between two variables, but including more and more variables in the analysis results in the errors. This is commonly known as the curse of dimensionality. Since we will use graphic methods, we will ignore this problem by now.

[1]Please contact me with any comments (typos, errors, unclear stuff, or other suggestions on how to improve these notes) at mbarron4 [at] ucsc

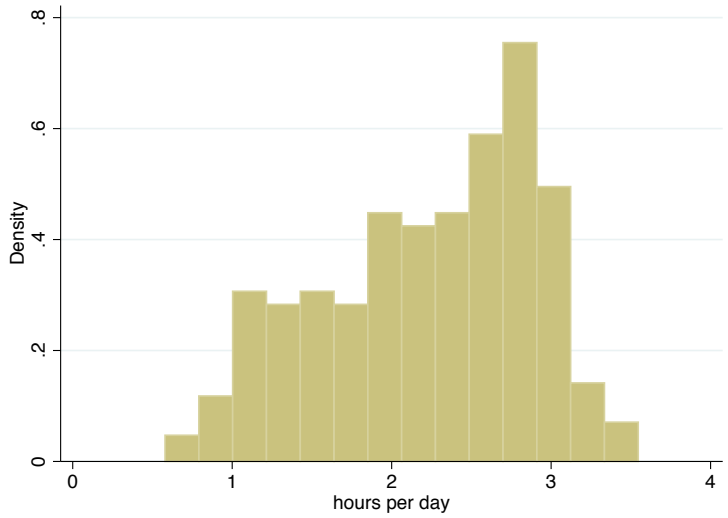
2 Density Estimation

2.1 Histogram

A histogram is a graphical representation of the distribution of a continuous random variable. To construct a histogram, we first split the data in intervals called bins, covering the entire range of the variable at hand. For instance, if the variable takes values from 0.5 to 3.5, the bins could be [0.5,1.0],]1.0,1.5],]1.5,2.0],]2.0,2.5],]2.5,3.0],]3.0,3.5]. Then, count the number of times the variable falls in each bin. Finally, draw a rectangle with base determined by the bin and height determined by the number of times the data falls in that bin.

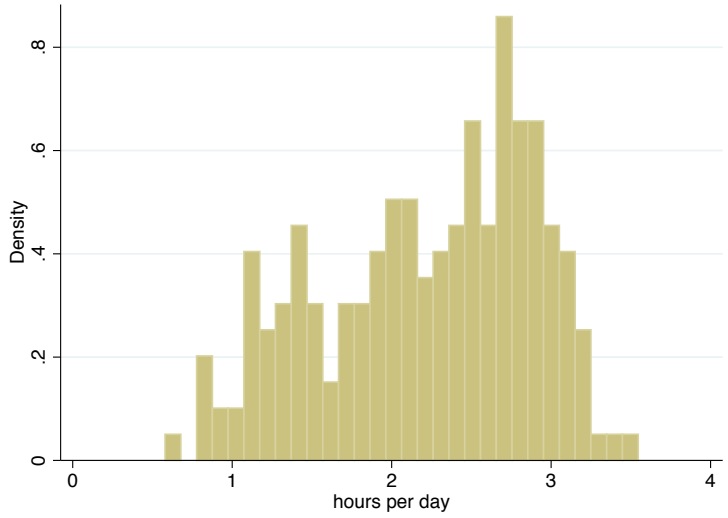
If we plot the histogram of "hours per day" in Stata, we will get

Figure 1 - Histogram



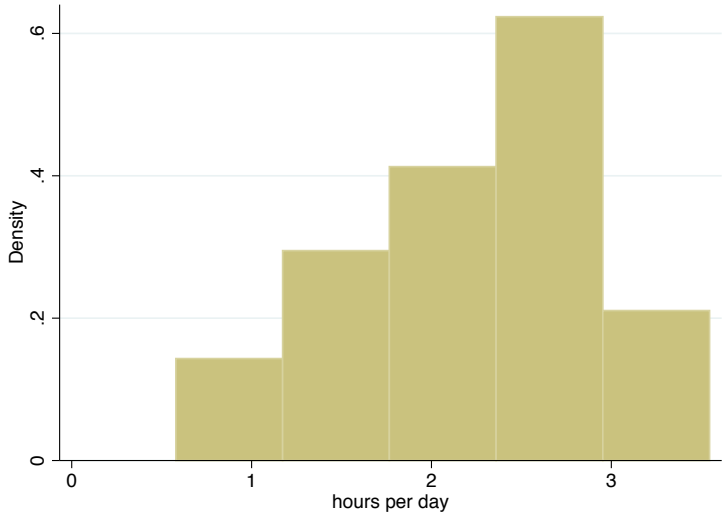
Note that the shape will depend on bin width: if we split the data in more bins than the "optimal" we'll get:

Figure 2 - Histogram with too many bins



If, on the other hand, we split the data in "too few" bins, we get a graph like the one below. The choice of bins is a little more advanced than we want to go, so for any practical purpose I would stick to Stata's default number of bins (unless you have a compelling reason to choose a particular bin width).

Figure 3 - Histogram with too few bins



do-file

```
clear all
set seed -7
set obs 200

cd [set your working directory]

gen x = 5*uniform()+5

gen y = 2 + sin(x) + 0.25*rnormal()

la var y "hours per day"
la var x "wage"

hist y, graphregion(color(white))
graph export "hist1.pdf", replace

hist y, bin(30) graphregion(color(white))
graph export "hist2.pdf", replace

hist y, bin(5) graphregion(color(white))
graph export "hist3.pdf", replace

hist y, graphregion(color(white))
graph export "kernel_1.pdf", replace
```

2.2 Kernel Density Estimation

Kernel Density Estimation is a method to estimate the probability density function of a random variable. Based on the observed sample, kernel density estimation allows to make inference about the variable distribution in the population. It can be thought of as a "smooth" version of the histogram.

Figure 4 - Kernel Density Estimate

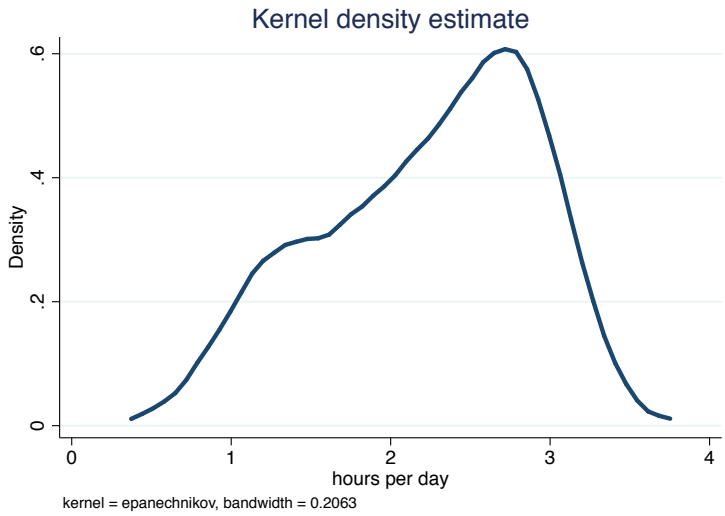
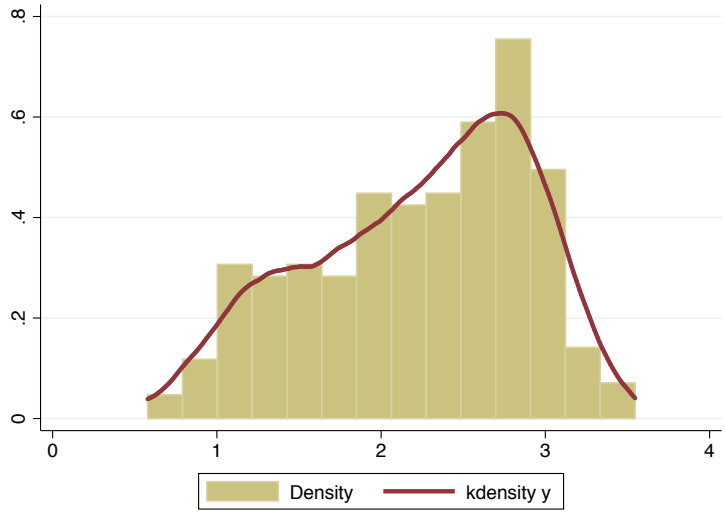
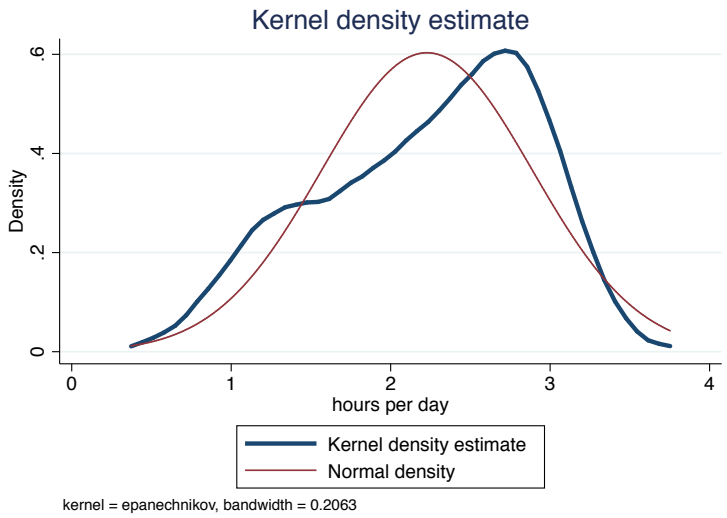


Figure 5 - Kernel Density and Histogram



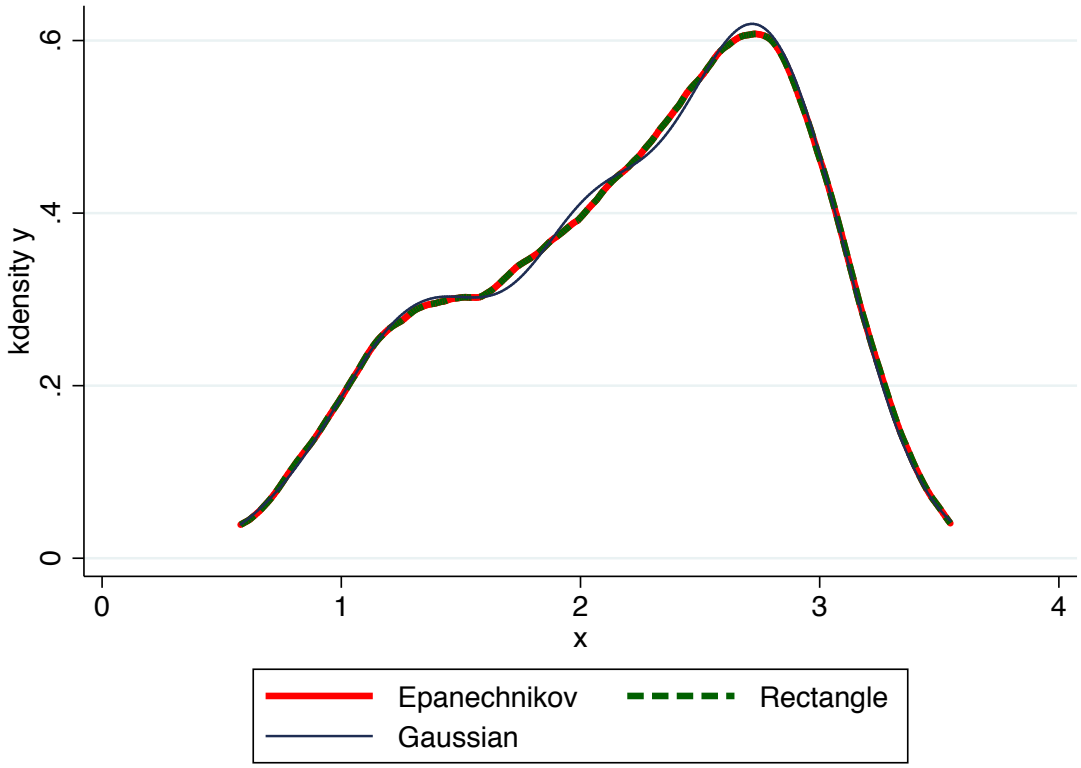
Including the "normal" option you can plot a normal density. This helps if you want to see if the variable at hand seems to follow a normal distribution.

Figure 6 - Estimated Kernel Density vs Normal Distribution



The choice of Kernel has very little impact on the density. The following graph shows the density resulting of using three different kernels: Epanechnikov, Rectangle, and Gausssian (a.k.a normal). This graph is larger than the others because the differences between the three lines are minimal. In fact, Epanechnikov and Rectangle lie on top of each other.

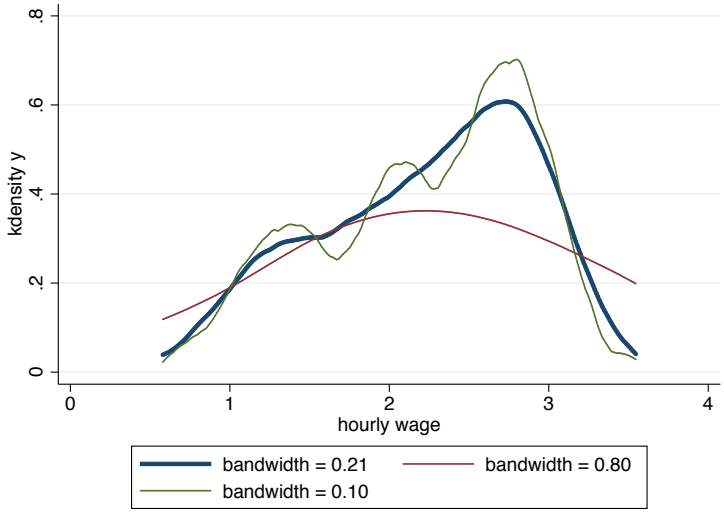
Figure 7 - Kernel Density Estimation with Different Kernel Functions



On the other hand, bandwidth is central to the shape of the density. There are many methods of optimal bandwidth choice, but this is an advanced topic. My recommendation is to simply use Stata's default optimal bandwidth (if you are interested, it is chosen by cross-validation).

Stata lets you choose a bandwidth different than the default. If you pick a narrow bandwidth (less than the optimal) you will produce an "undersmoothed" kernel. It is called under smoothed because it has many "jumps". Some of these jumps are present in this dataset, but not in the true population, so we should ignore them. Knowing what to ignore is tricky, by now just rely on Stata's optimal bandwidth.
If, on the other hand, you choose to wide a bandwidth, the resulting graph is "oversmoothed", which means it misses some of the most important features of the density

Figure 8 - Kernel Density Estimation with Different Bandwidths



.

do-file

```
twoway (hist y, graphregion(color(white))) ///
|| (kdensity y, graphregion(color(white)) lwidth(thick))
graph export "kernel_hist.pdf", replace

twoway (kdensity y, epan legend(label(1 Epanechnikov)) lcolor(red) lw(thick)) ///
|| (kdensity y, rectangle legend(label(2 Rectangle)) lpattern(dash) lw(thick) ///
lcolor(dkgreen)) ///
|| (kdensity y, gaussian legend(label(3 Gaussian)) lcolor(dknavy)) ///
, graphregion(color(white))
graph export "kernel_comparison.pdf", replace

twoway ///
|| (kdensity y, lwidth(thick) legend(label(1 "bandwidth = 0.21"))) ///
|| (kdensity y, bw(.80) legend(label(2 "bandwidth = 0.80"))) ///
|| (kdensity y, bw(.10) legend(label(3 "bandwidth = 0.10"))) ///
,  graphregion(color(white)) xtitle(hourly wage)

kdensity y, normal graphregion(color(white)) lwidth(thick)
graph export "kernel4.pdf", replace
```

3 Non-parametric Regressions

One of the most intuitive ways to transition from linear models to non-parametric models is with local linear regressions. In a nutshell, this method consists in running many linear regressions for different values of the covariate. This will become clearer in a second.

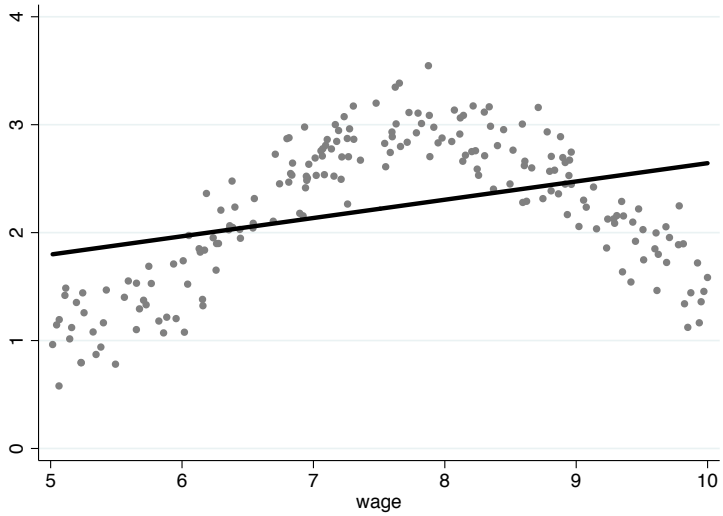
Imagine we have just received a dataset on labor supply (in hours per day) and wages (in US$ per hour) and we want examine the relation between both. Our fingers itch with anticipation, so we immediately run a regression of y on x and find the results shown in column 1 of Table 1. Things seem to be going pretty nice: the coefficient on x is positive and statistically significant at the 99% of confidence, (three stars, son!). Earning an additional dollar per hour is associated with working an additional 0.169 hours (10 minutes), so the effect is a bit on the smaller side but the coefficient makes sense (you didn't get economically insignificant results like 2 seconds or implausible ones, like 20 hours). Regression output will rarely look this good.

Table 1: Wages and Labor Supply, OLS

	(1) hours per day	(2) hours per day
wage	0.169***	0.806***
	(0.031)	(0.028)
high wage		12.489***
		(0.493)
wage x high wage		-1.588***
		(0.058)
Constant	0.950***	-3.127***
	(0.237)	(0.187)
Observations	200	200

Notes: wage: hourly wage in US$; high wage takes the value of 1 if hourly wage is greater than 8 US$ and zero otherwise. Standard errors in parenthesis. * p<0.10, **, p<0.05, *** p<0.01.

Figure 9 - Wage and Labor Supply, linear fit



However, Figure 9 shows that things aren't going so well with OLS. OLS provides the best linear fit, but the best linear fit is misleading in this case. There is a clear inverse-U relationship between x and y. Eyeballing it, the slope is positive for hourly wages between $5 and 8, and negative for hourly wages between $8 and 10. You can think of income effects coming at play for hourly wages above $8.

Given that we believe the relation to be positive from 5-8 and negative 8-10, we could include an interaction term in the regression to allow for a change in slope.

Figure 10 - Piecewise Linear OLS

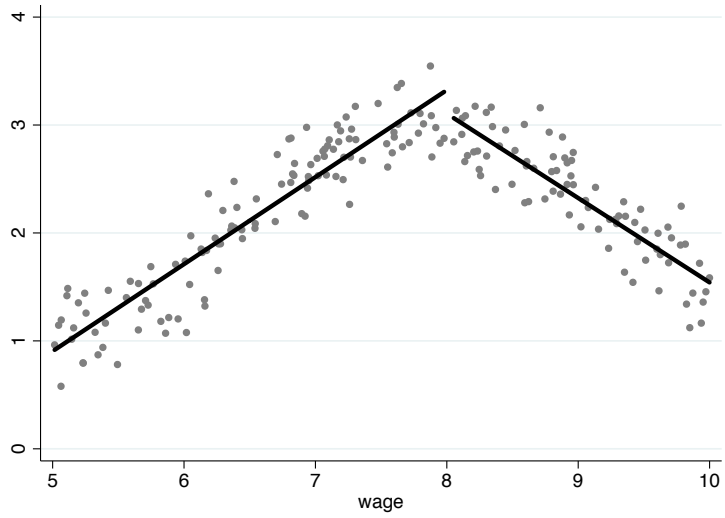
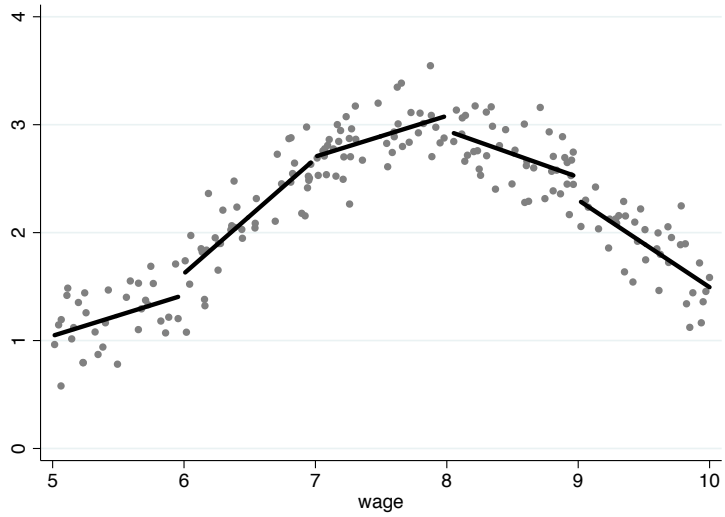


Figure 10 - Piecewise Linear OLS, with "smaller bandwidth"



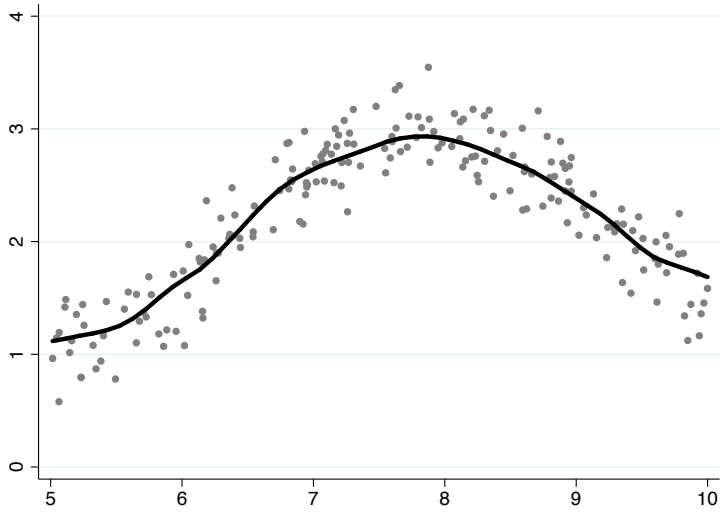
```
twoway (scatter y x, mcolor(gray) msize(small)) ///
|| (lfit y x, lcolor(black) lwidth(thick)) ///
, legend(off) graphregion(color(white))

twoway (scatter y x, mcolor(gray) msize(small)) ///
|| (lfit y x if x<8, lcolor(black) lwidth(thick)) ///
|| (lfit y x if x>=8, lcolor(black) lwidth(thick)) ///
, legend(off) graphregion(color(white))

twoway (scatter y x, mcolor(gray) msize(small)) ///
|| (lfit y x if x>=5 & x<6, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=6 & x<7, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=7 & x<8, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=8 & x<9, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=9 & x<10, lcolor(black) lwidth(thick))  ///
, legend(off) graphregion(color(white))
```

The above procedures may work if we are absolutely sure of where are the breaking points. But if we are not, we may want to use a non-parametric estimator, like local linear regressions. Local linear regression runs linear regressions *locally* meaning, in a neighborhood of x, i.e. within a given bandwidth. For instance, to estimate the slop at x=6, local linear regression takes all the data with x between 5.5 and 6.5, and estimates the slope at that point. Then, it moves to 6.1, takes all the points between 5.6 and 6.6 to estimate a new slope. Since both sets contain basically the same points, the slopes are going to be very similar, so the function looks continuous. Local polynomials (lpoly) goes a step further and includes polynomials in x to improve the estimation. Figure 11 shows the results

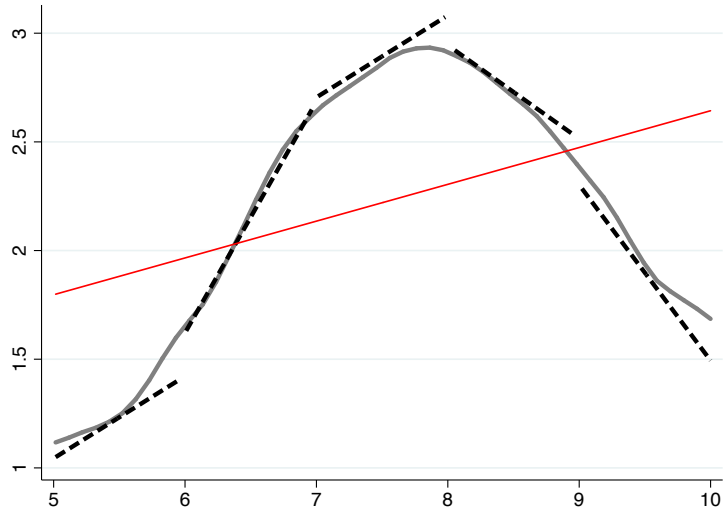
Figure 11 - Local polynomial estimators



do-file

```
twoway (scatter y x, mcolor(gray) msize(small)) ///
 || (lpoly y x, lcolor(black) lwidth(thick))  ///
, legend(off) graphregion(color(white))
```

Figure 12 - OLS, piecewise linear regression, Local Polynomials



do-file

```
twoway (lpoly y x, lcolor(gray) lwidth(thick) lpattern(dash)) ///
|| (lfit y x if x>=5 & x<6, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=6 & x<7, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=7 & x<8, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=8 & x<9, lcolor(black) lwidth(thick))  ///
|| (lfit y x if x>=9 & x<10, lcolor(black) lwidth(thick))  ///
|| (lfit y x, lcolor(red))  ///
, legend(off) graphregion(color(white))
```

4 Alternative Non-Parametric Regression Commands

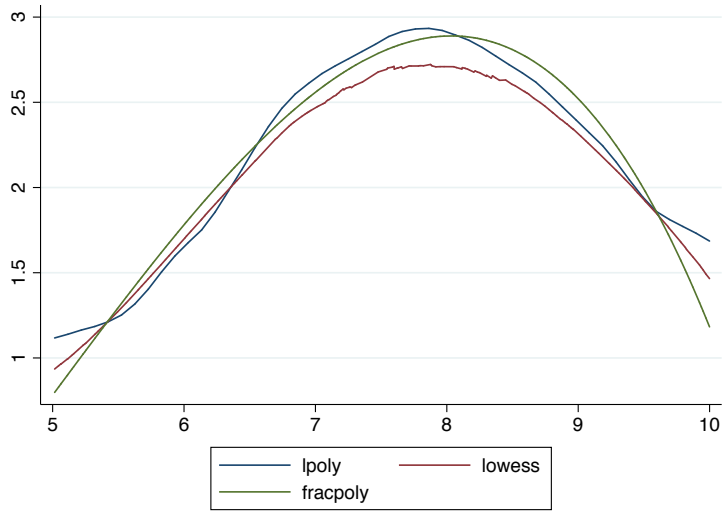
Together with poly, fpfit, lowess provide easy ways of estimating bivariate relations non-parametrically.

The options are pretty similar to those in *poly*. I will not cover them in lecture, but you may be asked to use them in the assignment or in the final exam.

The main difference between the methods seems to be at the extreme values of wages. This is because at wages close to 5 or 10, there are not many data points to the left (or the right), so the use of different weighting functions will likely produce slightly different results.

As with the choice of kernel, there is little difference in the method you use. The most important thing is the bandwidth used in the estimation. Choice of bandwidth is an advanced topic, so by now you should just use the default bandwidth chosen by Stata.

Figure 13 - Alternative Non-parametric Methods



do-file

```
twoway (lpoly y x, legend(label(1 "lpoly"))) ///
|| (lowess y x, legend(label(2 "lowess"))) ///
|| (fpfit y x, legend(label(3 "fracpoly"))) ///
, graphregion(color(white))
```

5 Confidence Bands

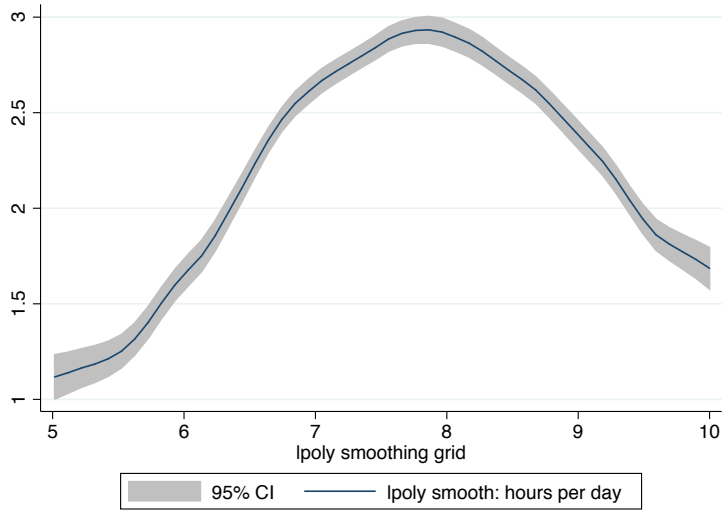
In a regression table, the point estimate doesn't tell the whole story. For instance, we need standard errors to build confidence intervals, which allow us to infer if a coefficient is significant or not. Similarly, in non-parametric analysis we can produce confidence bands.

An easy way of generating confidence bands is with the "ci" versions of lpoly and fpfit. For instance: lpolyci.

do-file

```
twoway (lpolyci y x), graphregion(color(white))
```

Figure 14 - Local Polynomials with Confidence Bands

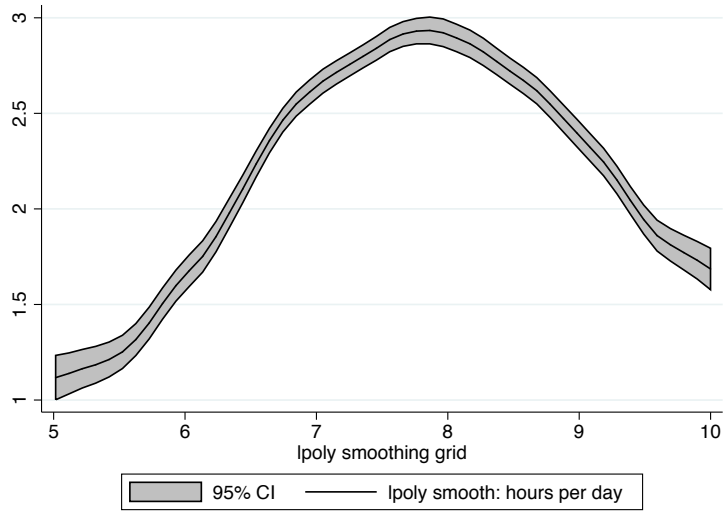


The above command allows us to create quick plots with confidence intervals. However, if we want to change some options we may run into a bit of trouble. For instance, if we change the line color to black, the resulting graph may not be what we were hoping for.

do-file

```
twoway (lpolyci y x, lcolor(black)), graphregion(color(white))
```

Figure 15 - Local Polynomials with Confidence Bands



The following code provides a way around this. We will first generate variables that contain the values of y and x in the graph, together with the standard errors. Then, we find the critical value of the test statistic to construct the upper and lower bound confidence intervals, and then we graph them as if they were lines.

do-file

```
lpoly y x, gen(xhat yhat) se(sehat) noscatter


* Replace 1.965 by the critical tstat

* upper bound, control:
g ub = yhat + 1.965*sehat

* lower bound, control:
g lb = yhat - 1.965*sehat

twoway (line yhat xhat, lcolor(dknavy) lwidth(thick)) ///
|| (line ub xhat, lcolor(black) lpattern(dash)) ///
|| (line lb xhat, lcolor(black) lpattern(dash)) ///
, ytitle("Y Axis Title") xtitle("X Axis Title") legend(off) ///
graphregion(color(white))
```

Figure 16 - Local Polynomials with Confidence Bands