Introduction to Nonparamteric Regression: Local Polynominal Regression & Loess

Youmi Suk

School of Data Science, University of Virginia

9. 21. 2021

Overview

Local Polynomial Regression

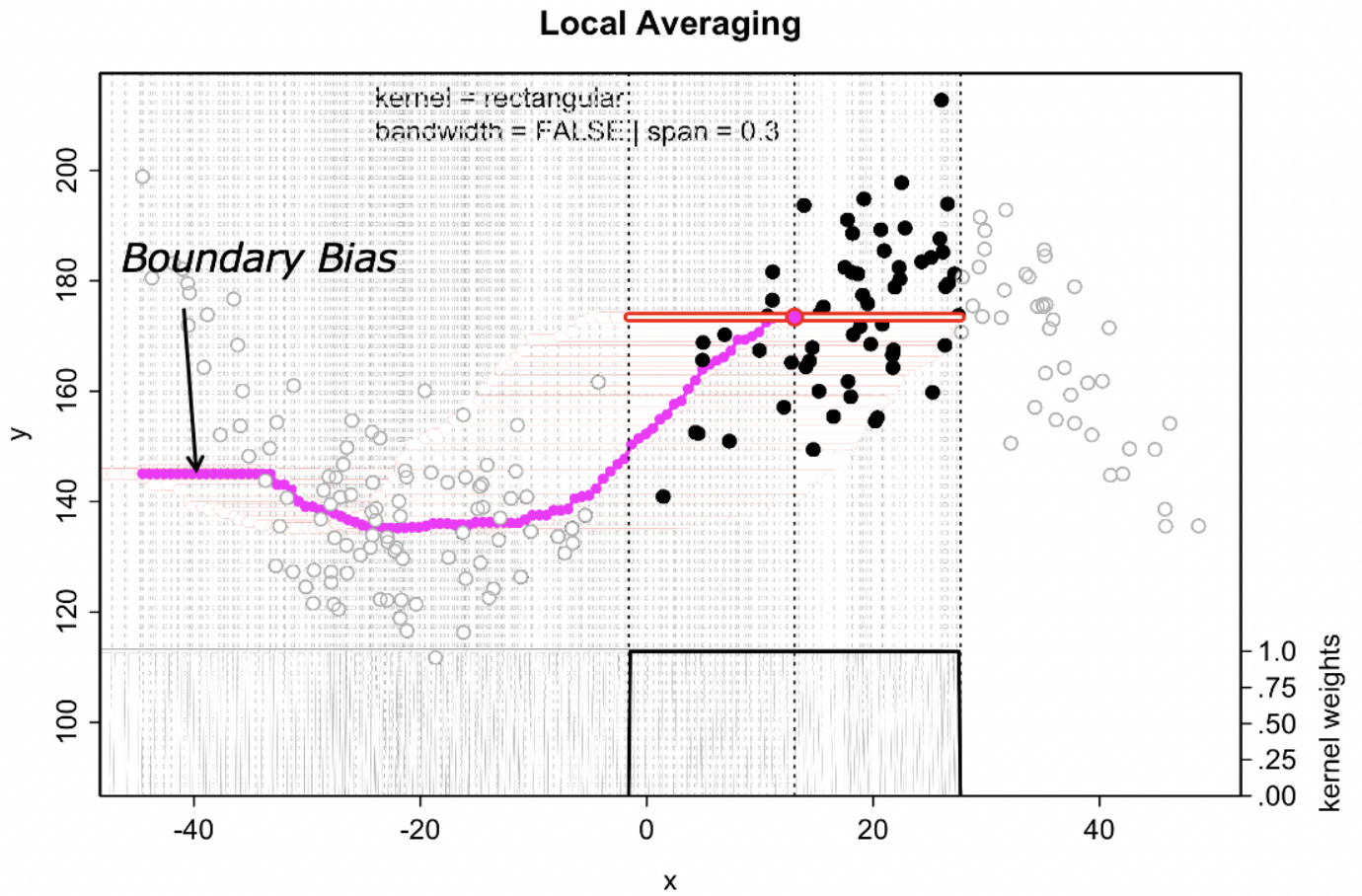
Locally Weighted Regression (LOESS/LOWESS)

Nonparametric vs. Parametric Regression

Regression/Scatterplot Smoothers

- Binning is a very rough method and depends, of course, on the location and size of the bins.
- An alternative approach are so called regression or scatterplot smoothers which construct a separate bin (=window) for each unique x-value and then calculates the mean (or other statistic) from the data within the window. The window is typically defined by kernel weights.
  - Local averaging & kernel estimation
  - Local polynomial regression
  - Locally weighted regression (loess/lowess)

Boundary Bias with Kernal Estimation



Local Polynomial Regression

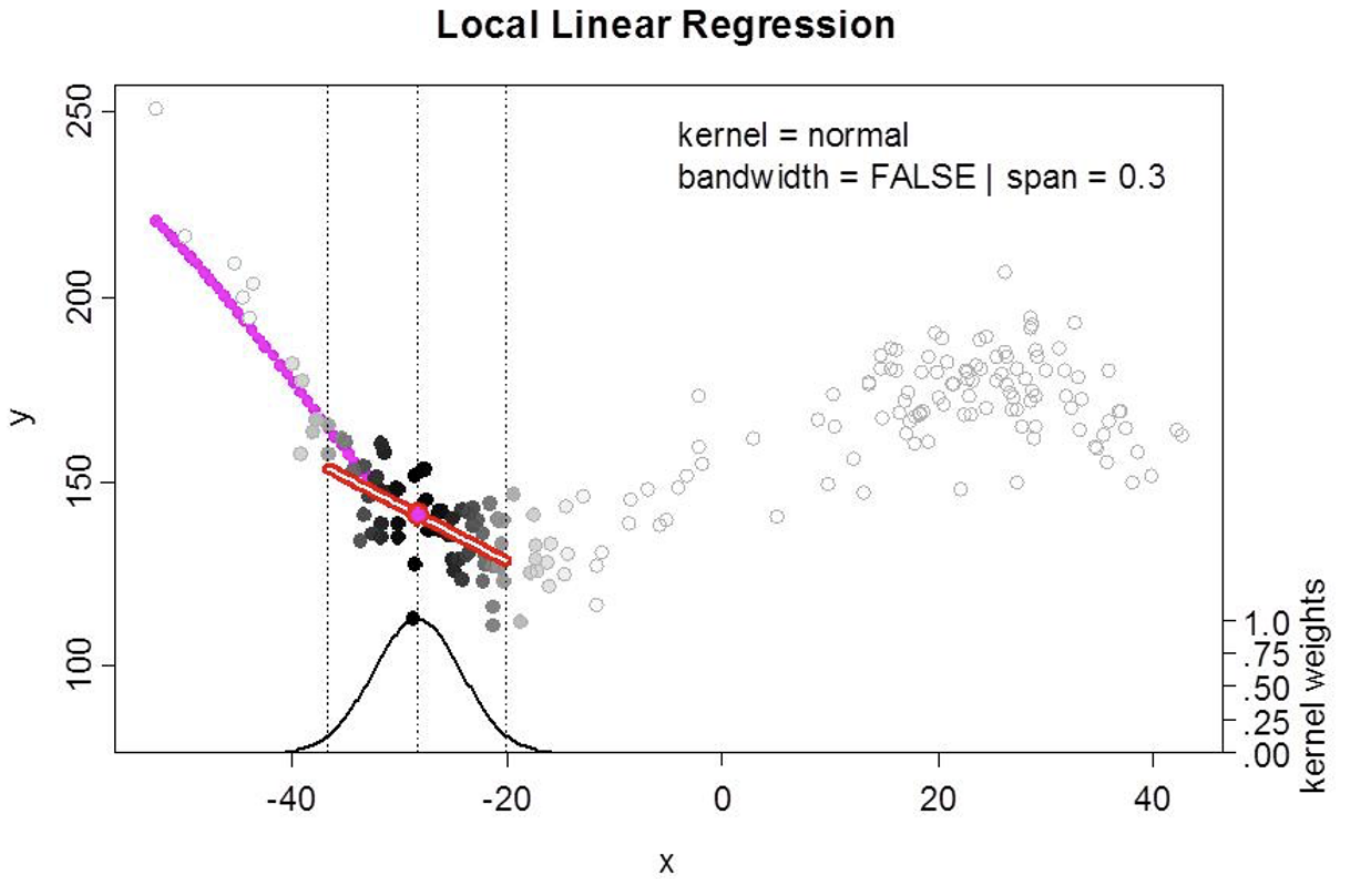
- Poor performance of local averaging and kernel estimation particularly at the boundaries of X
- Why not running a simple linear or quadratic regression inside each window?
- Reflects nonlinear relationships within windows.
- Helps in "extrapolating" to boundaries
  - E.g., local quadratic regression: run a quadratic regression with (kernel) weights

$$\hat{Y}_i = A + B_1(x_i - x_0) + B_2(x_i - x_0)^2$$

where the estimated mean value at x_0 is given by the estimated intercept A: $\hat{Y}|x_0 = A$.

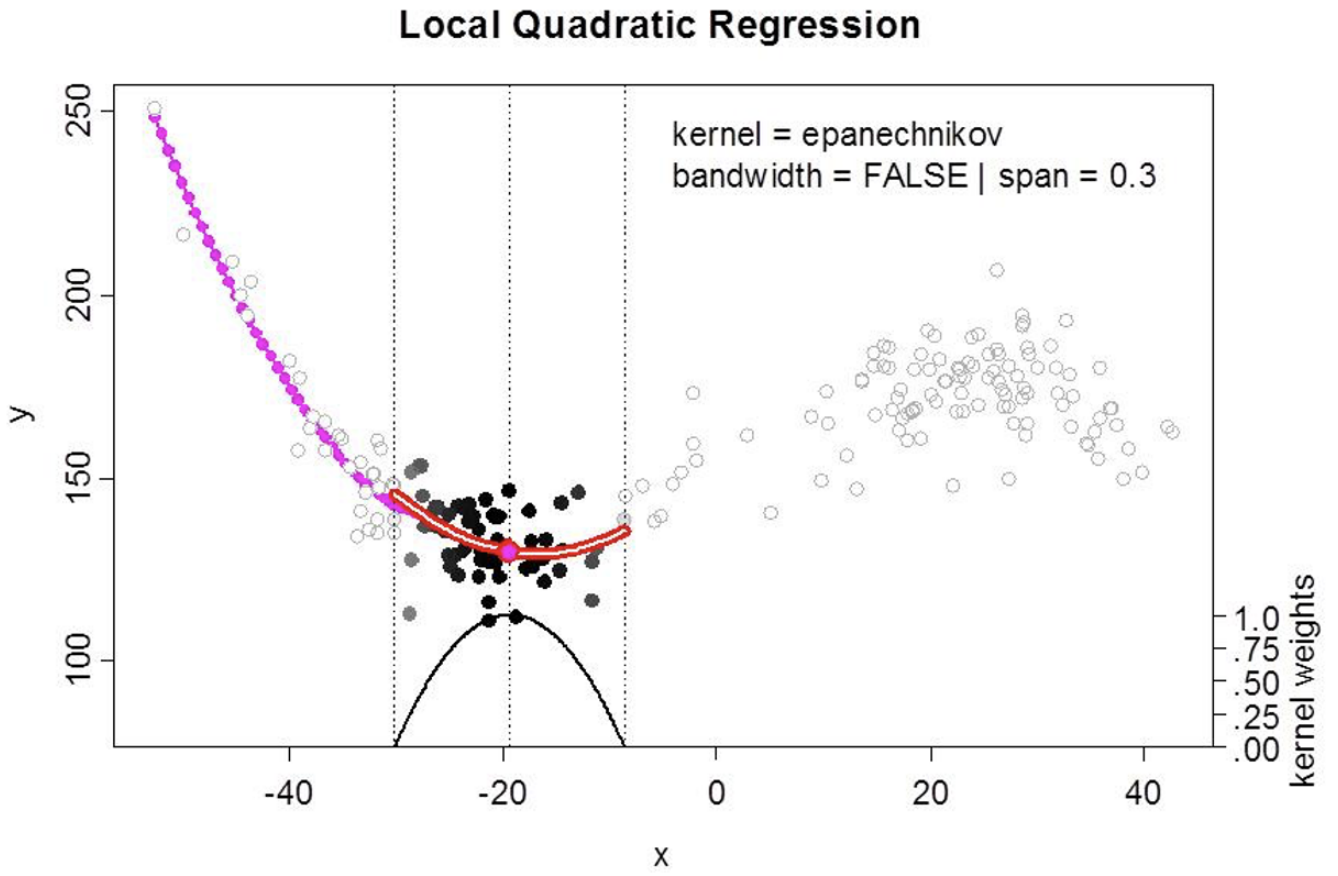
Local Linear Regression

- with a normal kernal



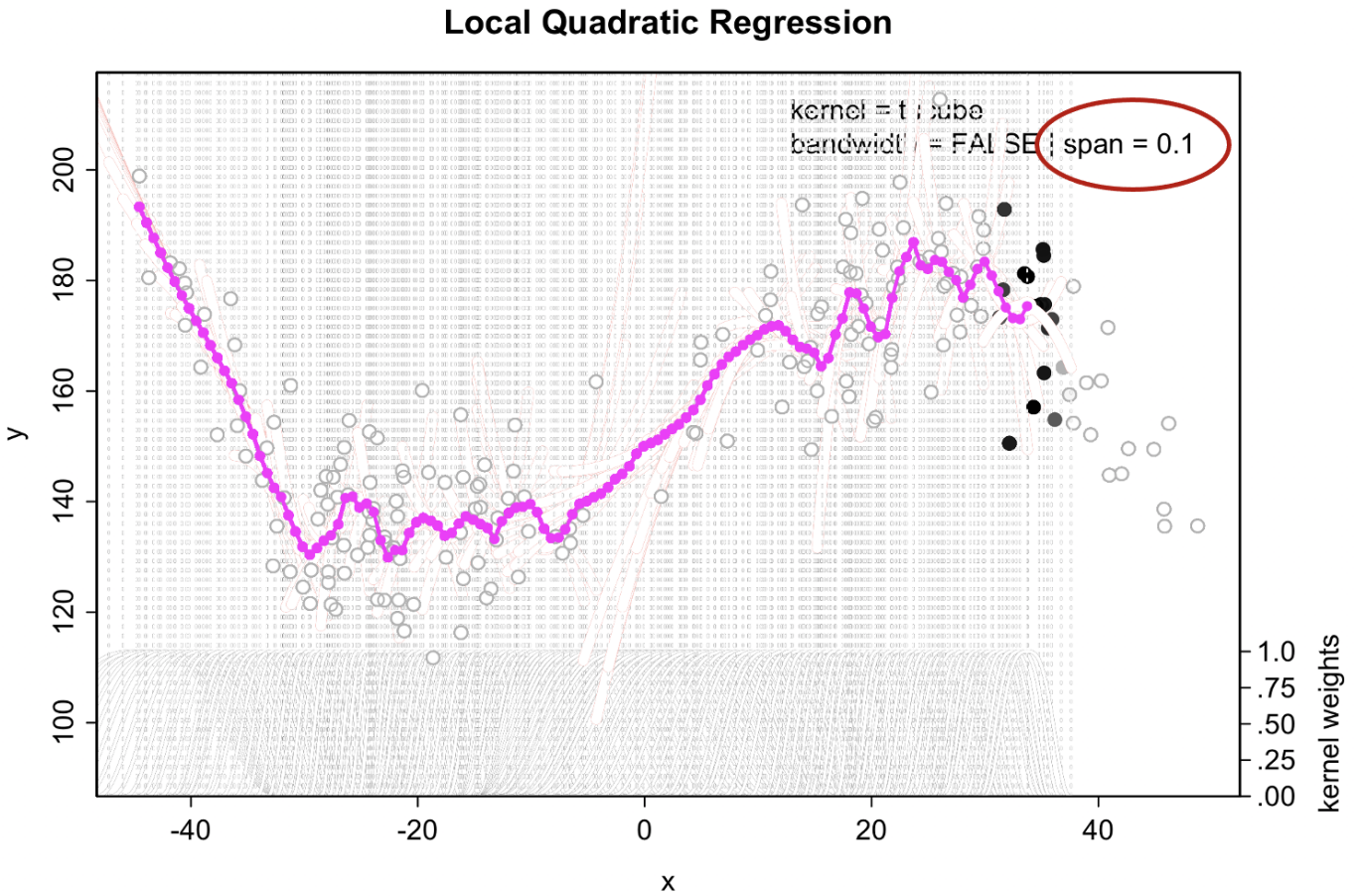
Local Quadratic Regression

- with an Epanechnikov kernal



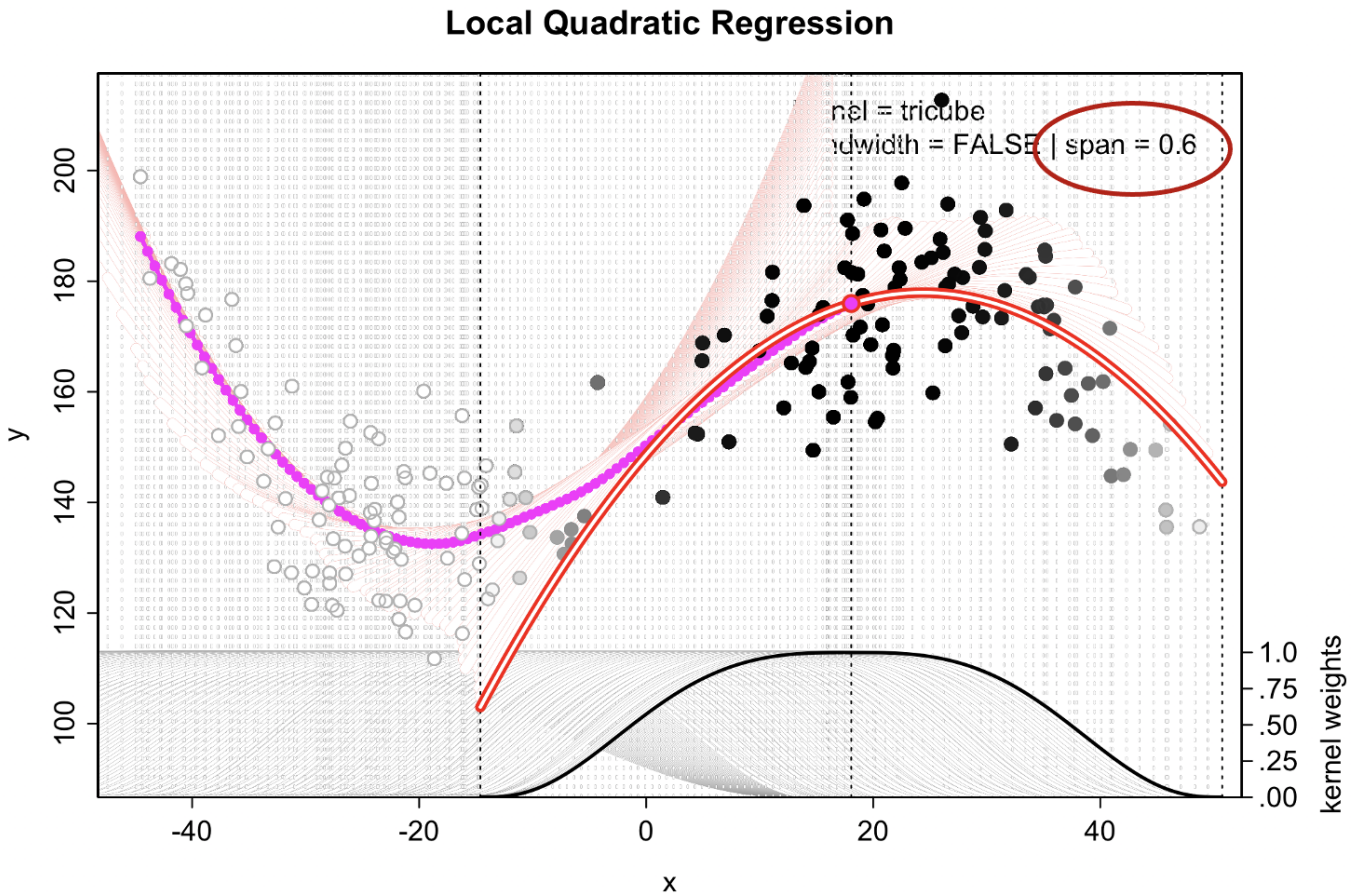
Local Quadratic Regression

- with a small span



Local Quadratic regression

- with a large span



How to do it in R?

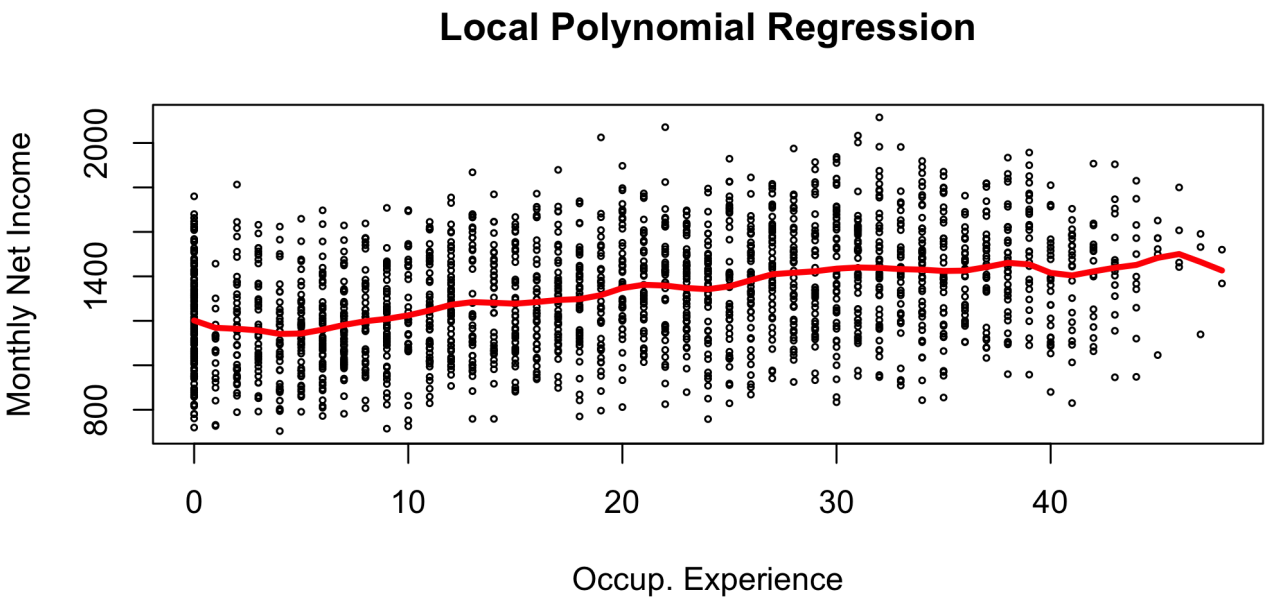
- Write our own local linear or quadratic regression function: `k.reg()`

```
k.reg <- function(y, x, h, deg = 1)
{
   # computes a predicted value of y for each unique value of x
   # with a centered normal kernel
   # y   ... vector of dependent variable
   # x   ... vector of independent variable
   # h   ... bandwidth (standard deviation of the normal kernel
   # deg ... degree of polynomial (1: linear, 2: quadratic)

   x.val <- sort(unique(x))
   avrg <- function(y, x, x.loc, h, deg) {
      x.c <- x - x.loc         # centering
      wt <- dnorm((x.c) / h)   # weights
      if(deg == 1) m <- lm(y ~ x.c, weights = wt)$coef[1] # extracts
      if(deg == 2) m <- lm(y ~ x.c + I(x.c^2), weights = wt)$coef[1]
      m                        # return mean value
   }
   m.vec <- sapply(x.val, avrg, y = y, x = x, h = h, deg = deg)
   cbind(x = x.val, y = m.vec)
}
```

Example: Local polynomial regression

- with function `plot`

```
out.kreg <- with(incex, k.reg(income, oexp, 1))              # get x ar
plot(income ~ oexp, data = incex, cex = .4, xlab = 'Occup. Experience
     ylab = 'Monthly Net Income', main = 'Local Polynomial Regressior
lines(out.kreg, col = 'red', lwd = 3)                        # add path
```



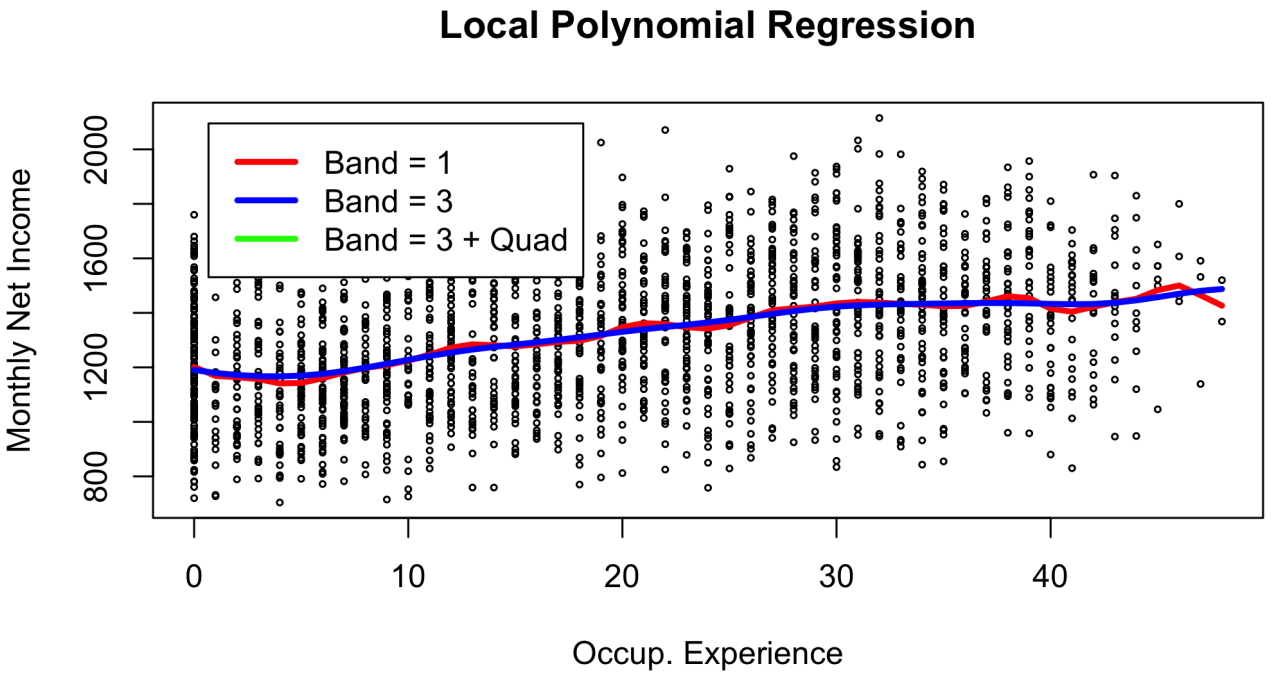
Example: Local polynomial regression

- with function `ggplot`

```
ggplot(incex, aes(x=oexp, y=income)) + geom_point() +
  labs(x = 'Occupational Experience', y = 'Monthly Net Income', title
  geom_line(data=out.kreg, aes(x=x, y=y), col = "red", size=1) + ther
```

Example: Local polynomial regression

- varying bandwidth and degrees with function `plot`



Example: Local polynomial regression

- varying bandwidth and degrees with function `ggplot`

Local/Locally Weighted Regression (LOESS/LOWESS)

LOESS/LOWESS is an improved version of local polynomial regression (with a normal kernel) which is less sensitive to outliers

```
loess(formula, data, weights, subset, na.action, span = 0.75,
  degree = 2, ...)
```

```
span ... span of window; default is 75% of observ.
degree ... degree of polynomial; default is
         a quadratic polynomial (= 2)
```

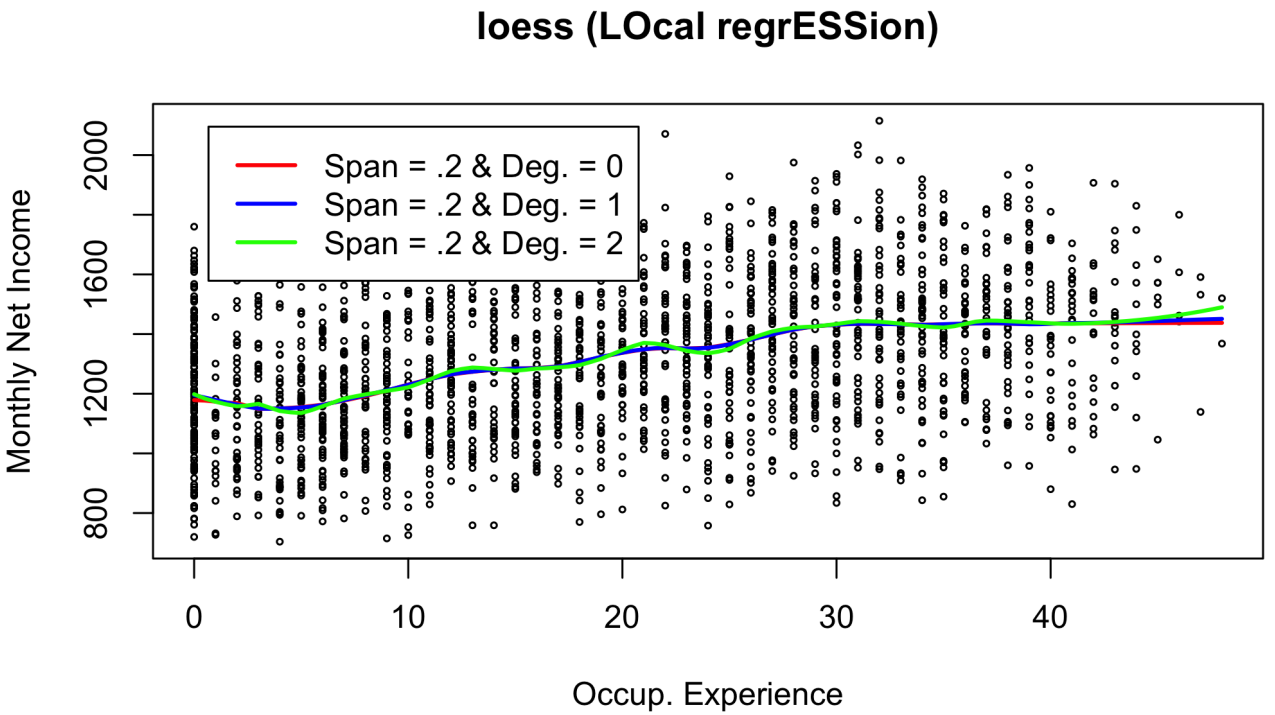
How to do it in R?

- with varying degrees

```
plot(income ~ oexp, data = incex, cex = .4, xlab = 'Occup. Experience
     ylab = 'Monthly Net Income', main = 'loess (LOcal regrESSion)')

x.val <- seq(0, 48, by = .5)

# ::::: local regression (loess): but with simple means since degree
out.lss <- loess(income ~ oexp, data = incex, span = .2, degree = 0)
y.pred <- predict(out.lss, data.frame(oexp = x.val))  # get predicted
lines(x.val, y.pred, col = 'red', lwd = 2)

# ::::: local LINEAR regression (loess): degree = 1 :::::
out.lss1 <- loess(income ~ oexp, data = incex, span = .2, degree = 1)
y.pred <- predict(out.lss1, data.frame(oexp = x.val))
lines(x.val, y.pred, col = 'blue', lwd = 2)

# ::::: local QUADRATIC regression (loess): degree = 2 :::::
out.lss2 <- loess(income ~ oexp, data = incex, span = .2, degree = 2)
y.pred <- predict(out.lss2, data.frame(oexp = x.val))
lines(x.val, y.pred, col = 'green', lwd = 2)

legend('topleft', c('Span = .2 & Deg. = 0', 'Span = .2 & Deg. = 1',
```

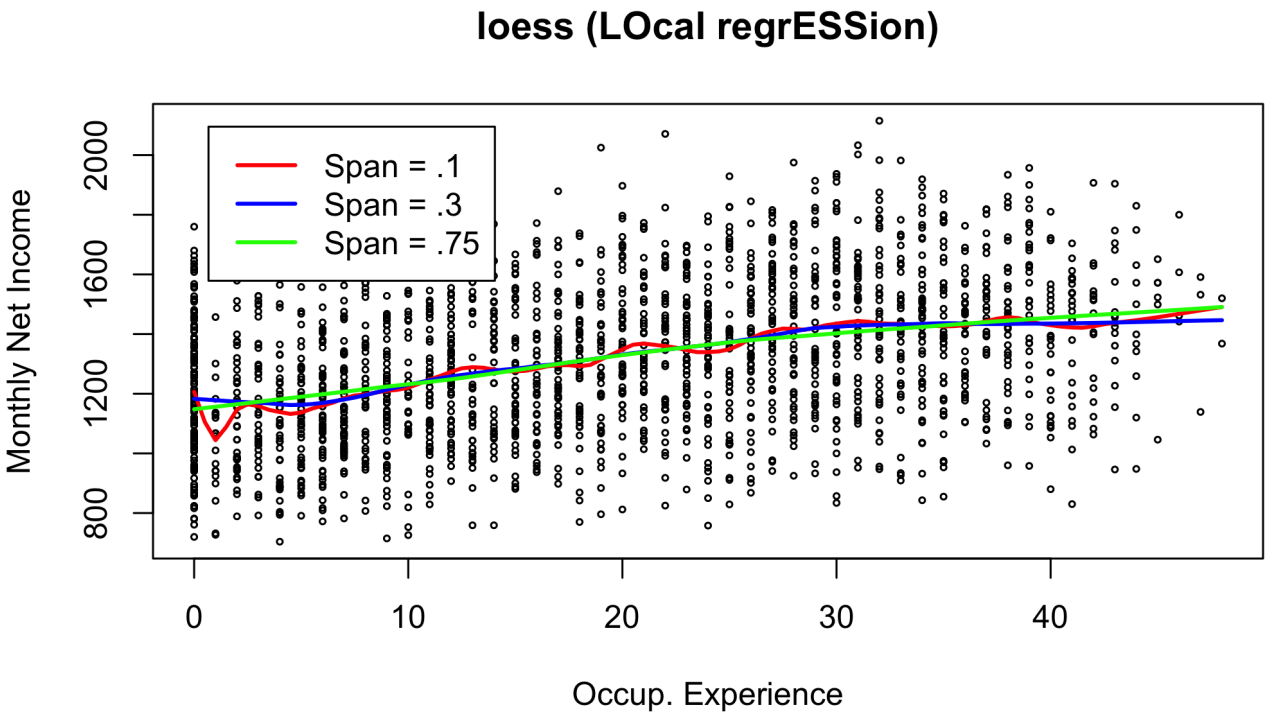
How to do it in R?

- with varying degrees



How to do it in R?

- with varying spans



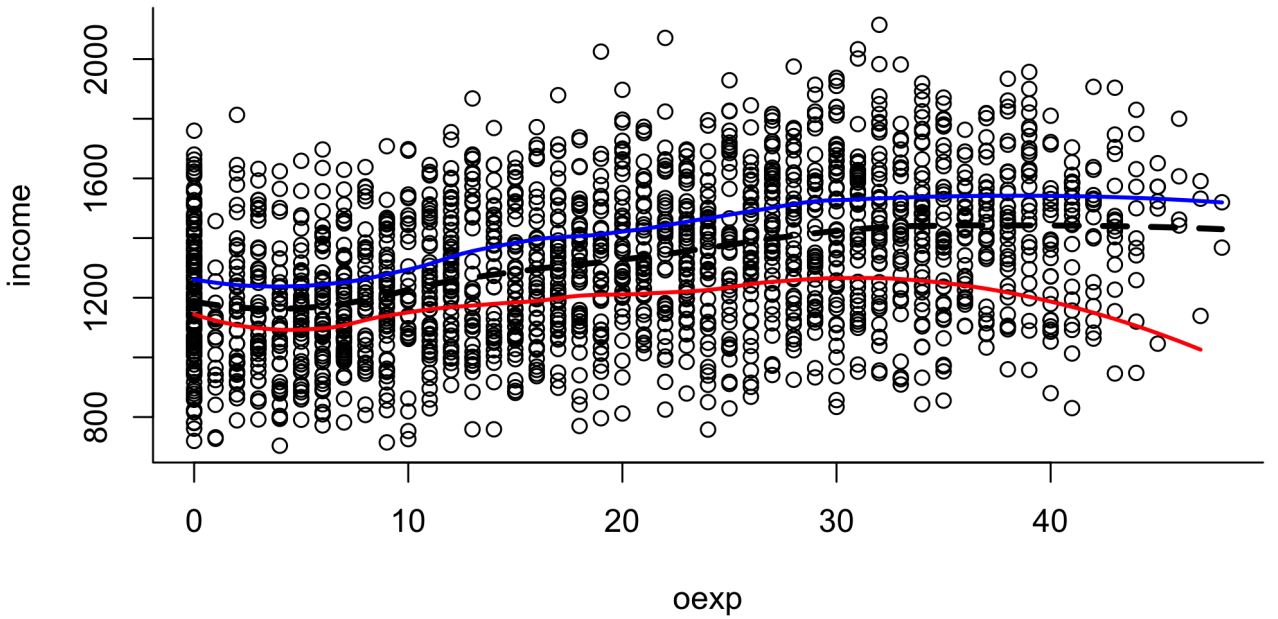
How to do it in R?

- separate loess lines

```
# loess() for overall sample, men, and women
plot(income ~ oexp, data = incex, bty = 'L')
out.all <- loess(income ~ oexp, incex, span = .5, degree = 2)
out.m <- loess(income ~ oexp, incex,
  subset = sex == 'male', span = .5, degree = 2)
out.f <- loess(income ~ oexp, incex,
  subset = sex == 'female', span = .5, degree = 2)
x.val <- seq(0, 48, by = .5)

lines(x.val, predict(out.all, x.val), lwd = 3, lty = 2)
lines(x.val, predict(out.m, x.val), col = 'blue', lwd = 2)
lines(x.val, predict(out.f, x.val), col = 'red', lwd = 2)
```

How to do it in R?

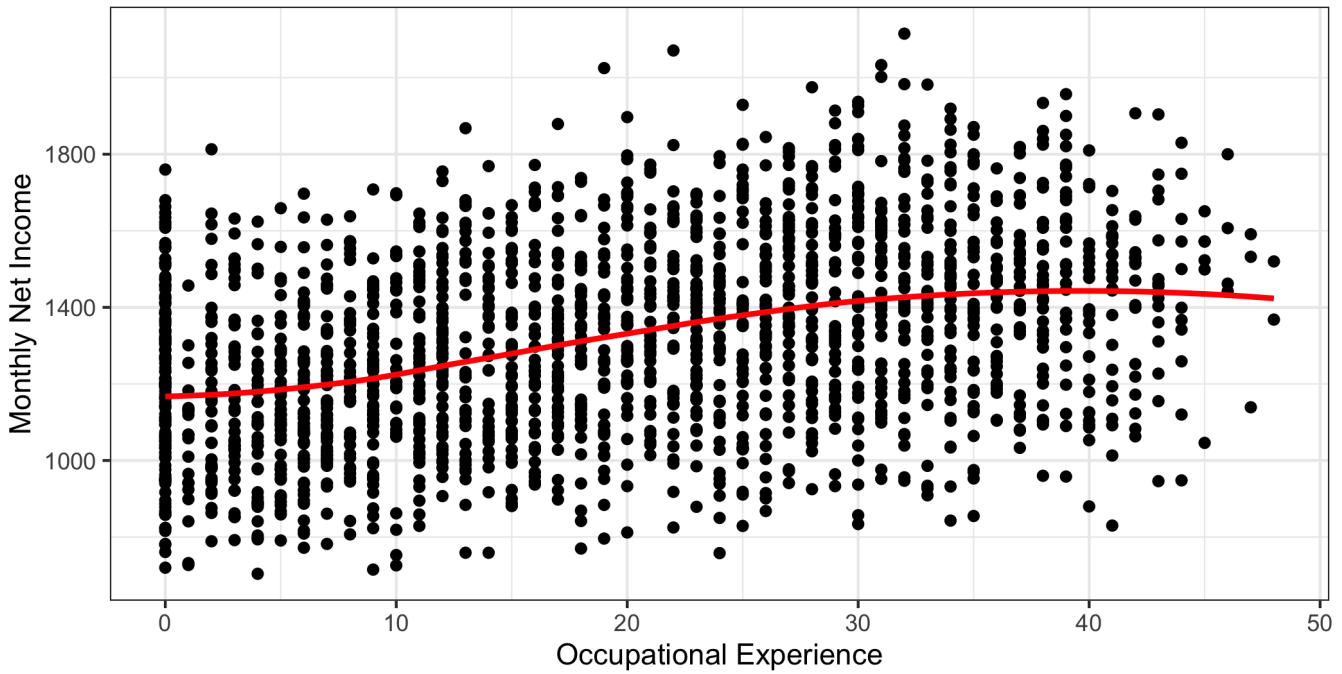
- separate loess lines



function geom_smooth in ggplot2

- default span = 0.75

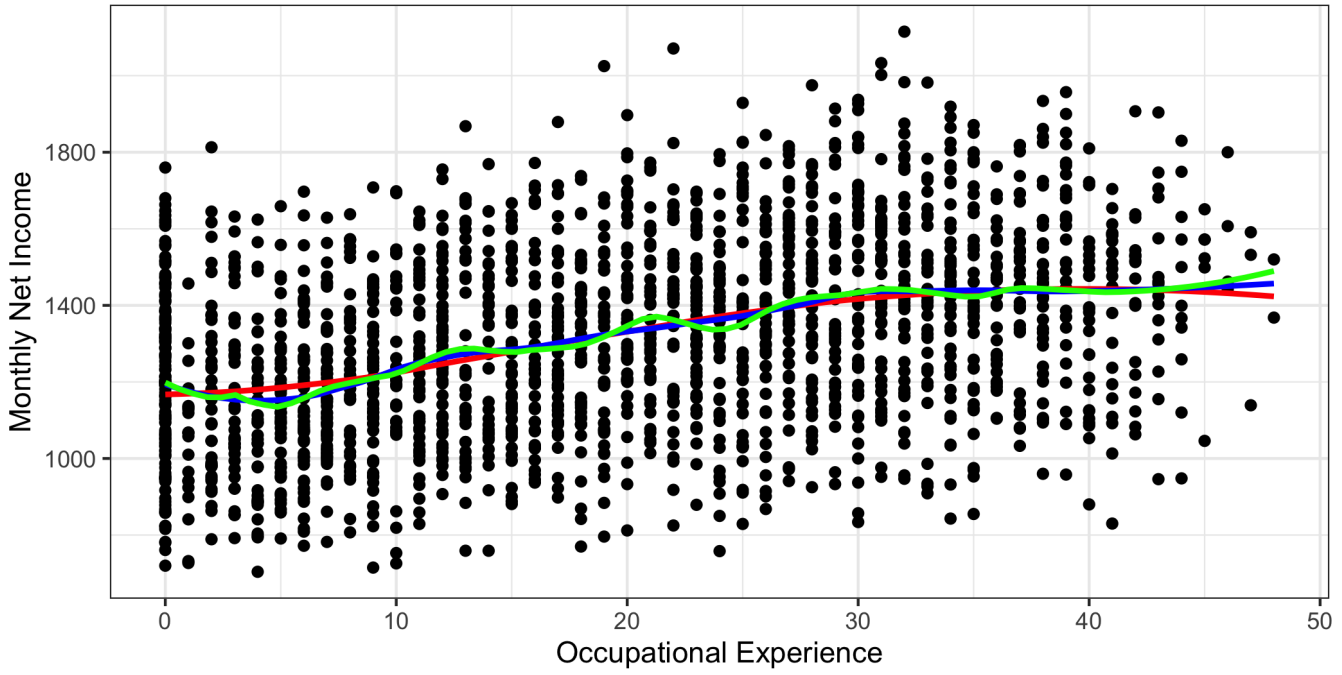
```
ggplot(incex, aes(x=oexp, y=income)) + geom_point() + labs(x = 'Occup
        geom_smooth(method='loess', formula= y~x, col="red", se = FAl
```



function geom_smooth in ggplot2

- try different spans

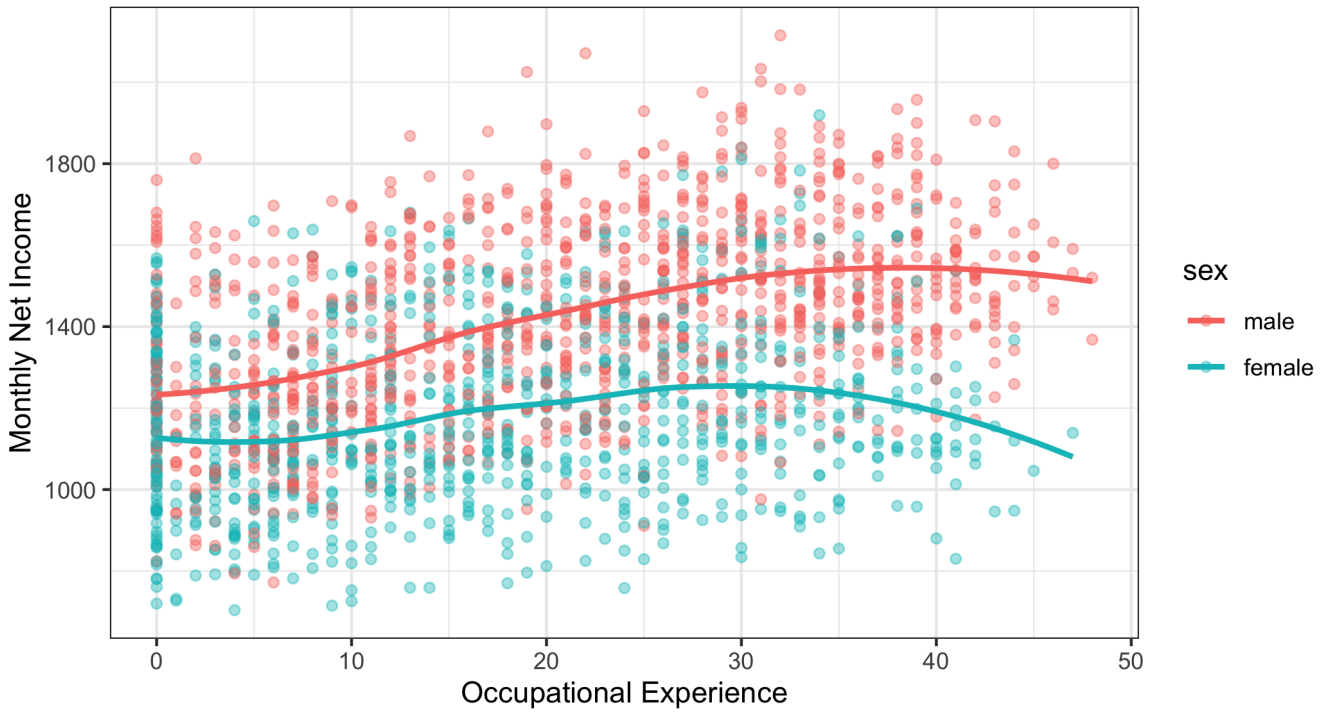
```
ggplot(incex, aes(x=oexp, y=income)) + geom_point() + labs(x = 'Occup
        geom_smooth(method='loess', formula= y~x, span=0.75, col="red
        geom_smooth(method='loess', formula= y~x, span=0.4, col="blue
        geom_smooth(method='loess', formula= y~x, span=0.2, col="gree
```



function geom_smooth in ggplot2

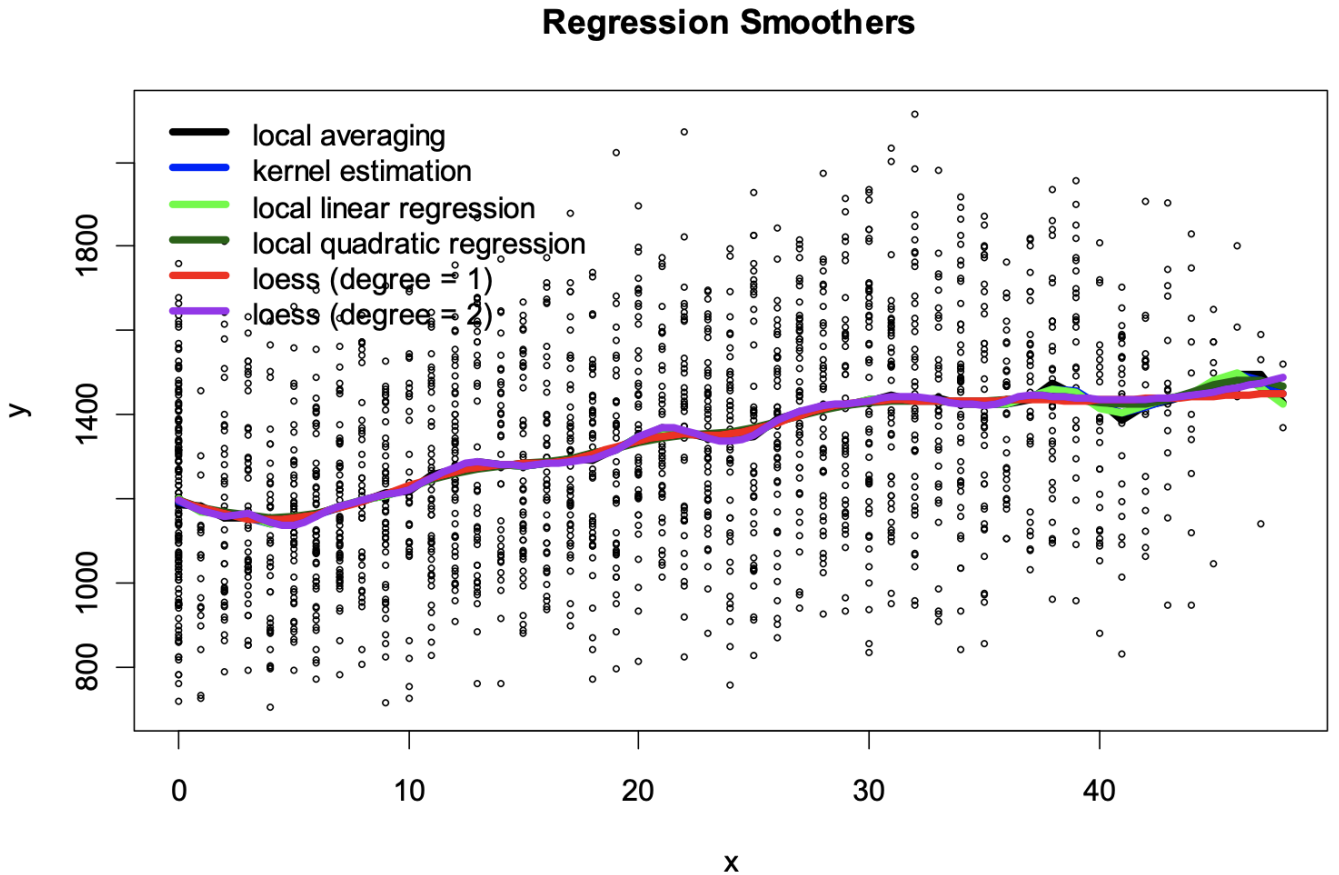
- with a color separator

```
ggplot(incex, aes(x=oexp, y=income, col=sex)) + geom_point(alpha=0.4)
        geom_smooth(method='loess', formula= y~x, se = FALSE) + theme
```



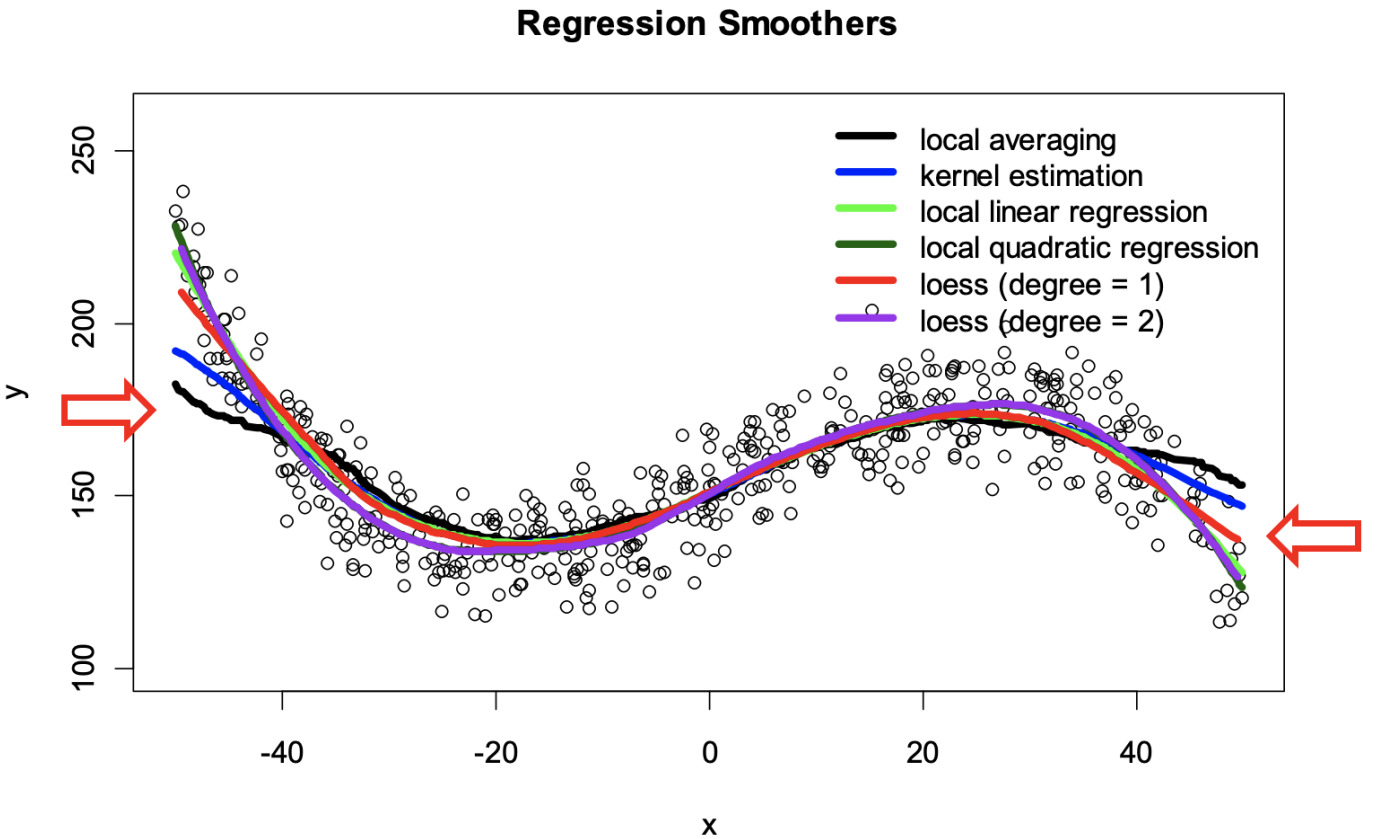
Comparison of Regression Smoothers

- with empirical data



Comparison of Regression Smoothers

- with simulated data



Regression Smoothers: Summary

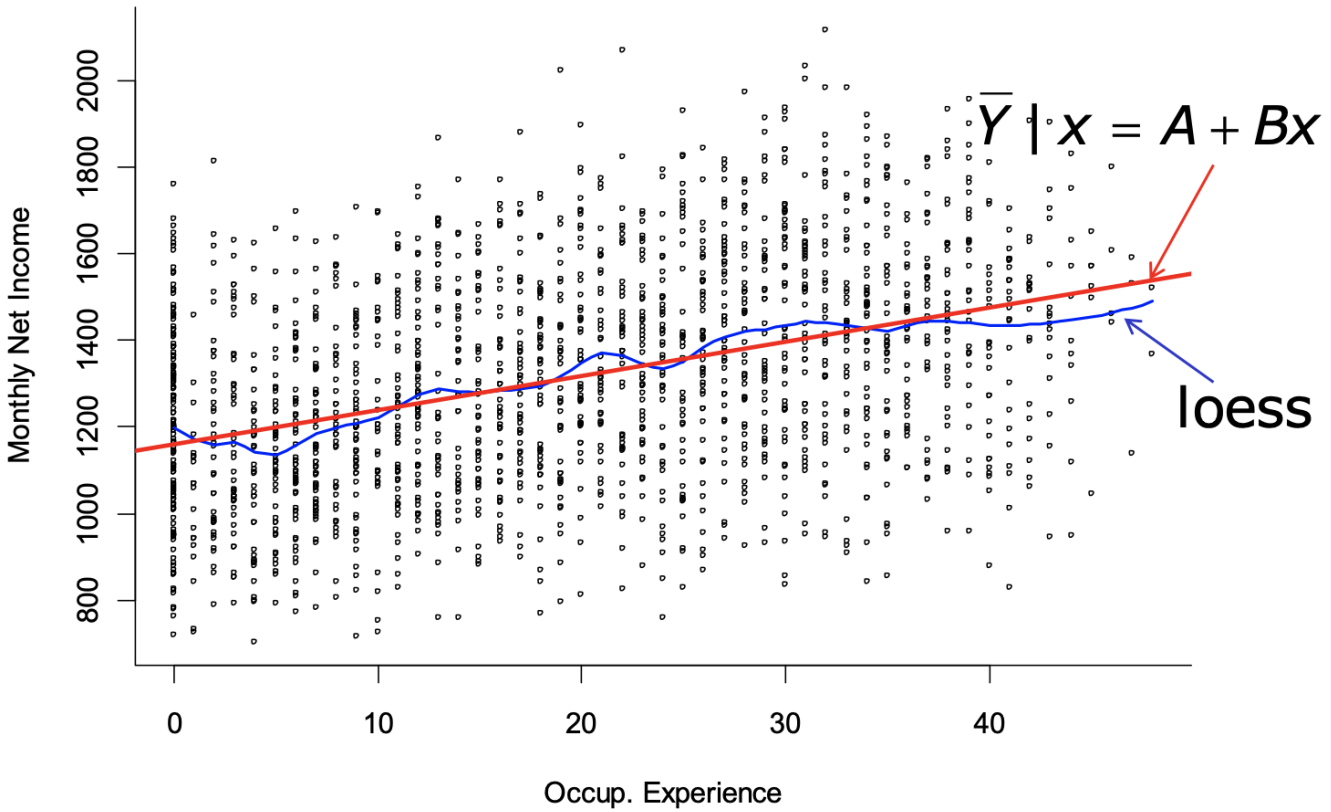
- Local averaging and kernel estimation perform rather poorly at the boundaries (boundary bias) and whenever the functional form is highly nonlinear (within windows).
- Performance of local polynomial regression and LOESS is very similar In any case, degree of smoothing depends on the window widths (as defined by the bandwidth or span)
- In practice, you can either use the `loess()` function or the local polynomial regression function `locpol()` from the locpol package.

Regression Smoothers: Summary

- Nonparametric regression works well for a single explanatory variable; though it is easy to generalize nonpar. regression to more than one explanatory variable, it is hard to do so in practice (would need huge sample sizes & is computationally expensive)
- However, the relationship between an outcome (Y) and explanatory variable (X) can frequently be described by a parametric model.
- For instance, if the linearity assumption is reasonable we can use a linear model for describing the relation between Y and X.

Nonparametric vs. Parametric Regression

- Nonparametric vs. linear path of means: linearity assumption seems plausible.



Nonparametric vs. Parametric Regression

- By using a parametric model we typically lose some local information.
- Instead of using the mean value for each specific x-value (around 50 means; one for each year of occupational experience) we now reduce the path of means to only two parameters: the intercept A and slope B.
- However, if the linearity assumption is wrong the linear (parametric) path of means is misleading!

Linear Regression

A simple linear model is wrong, and it needs a polynomial (cubic) model—both belong to the class of linear regression models.