Brief Introduction to Nonparametric Regression

- Scatterplot & Conditional Means (Path of Means)
- Binning
- Local Averaging
- Kernel Estimation
- Local Polynomical Regression
What is Regression Analysis

Regression analysis traces the conditional distribution of $Y$—or some aspect of this distribution, such as its mean—as a function of explanatory variables $X_1, \ldots, X_k$, i.e., how does the distribution/mean of $Y$ change as $X$ varies?

Conditional distribution

$$p(Y \mid x_1, x_2, \ldots, x_k) = f(x_1, x_2, \ldots, x_k)$$

Conditional expectation

$$E(Y \mid x_1, x_2, \ldots, x_k) = f(x_1, x_2, \ldots, x_k)$$
Regression Analysis: Example

Scatterplot: *Income* by *Occupational Experience*

*Income ~ Occupational Experience*
Conditional Distribution: Example

Cond. distribution of *Income* given *Occ. Experience*

\[ \hat{\rho}(\text{Income} \mid \text{Occ. Experience}) = f(\text{Occ. Experience}) \]
Conditional Expectation: Example

Cond. expectation (= cond. mean for observed data)

\[ E(\text{Income} | \text{Occ. Experience}) = f(\text{Occ. Experience}) \]
How to Do it in R?

```r
> attach(incex)
> plot(income ~ oexp, data = incex, cex = .4)
> # means
> m.vec <- tapply(income, oexp, mean)
> o.vec <- sort(unique(oexp))
> points(o.vec, m.vec, col = 'red', pch = 16, type = 'o', lwd = 2)
> detach(incex)
```
Nonparametric Regression

We investigate the conditional distribution/mean of a response variable without making any assumptions about the functional form or the shape of the distribution → nonparametric regr.

- With large data sets and discrete explanatory variable(s) like age in years or educational level, we can estimate the conditional distribution/mean of $Y$ directly

- If the explanatory variables are continuous (age measured in days, achievement scores) we don’t have enough observations for each unique value; thus we aggregate the explanatory variable into a large number of narrow bins and take the mean value (or other statistics like quantiles) → binning
Binning & Path of Means

5-year bins: \( C \in \{0-4, 5-9, \ldots, 45-49 \ \text{year}\} \)
conditional mean given bins \( C \):
Binning & Path of Means

Conditional means as a step-function

Binning

P. M. Steiner
How to Do it in R?

```r
> incex$oexp10 <- cut(incex$oexp, breaks =
  seq(0, 50, by = 5), right = F)  # creates a "factor"
> m.vec <- with(incex, tapply(income, oexp10, mean))
> o.vec <- with(incex, tapply(oexp, oexp10, mean))

> # Plot with path of means
> plot(income ~ oexp, data = incex, cex = .4)
> lines(o.vec, m.vec, type = 'o', col = 'red', lwd = 2)
> abline(v = seq(4.5, 50, by = 5), lty = 2)

> # Plot with step-function of means
> plot(income ~ oexp, data = incex, cex = .4)
> points(seq(-.5, 50, by = 5), c(m.vec, m.vec[length(m.vec)]), type = 's', col = 'red', lwd = 3)
```
Regression/Scatterplot Smoothers

Binning is a very rough method and depends, of course, on the location and size of the bins (as it is the case with histograms)

An alternative approach are so called regression or scatterplot smoothers which constructs 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)
Local Averaging

Local averaging calculates the mean (or other statistic) for a window centered at a specific value \( x_0 \)

Procedure for local averaging:
1. Define value \( x_0 \) for which you want to calculate the mean
2. Construct a symmetric neighborhood around \( x_0 \) (=window): choose a window width using either the
   - bandwidth (=half the window width in absolute measurement units of \( x \)) or
   - span (=a fraction of the data covered by the window, e.g., .3 = 30%)
3. For observ. within the window, compute the mean of \( Y \)
4. Do so for all other \( x \)-values
Local Averaging

More formally, the weighted local average for a given value $x_0$ is given by

$$\bar{Y} \mid x_0 = \frac{\sum_{i=1}^{n} w_i Y_i}{\sum_{i=1}^{n} w_i}$$

where weight $w_i = 1$ if observation $i$ lies within the window and $w_i = 0$ if it is outside the window:

$$w_i = \begin{cases} 1 & \text{for } |(x_i - x_0)/h| < 1 \\ 0 & \text{for } |(x_i - x_0)/h| \geq 1 \end{cases}$$

$h$ is the bandwidth of the window (half of the window width). Note that the weight function is identical to a rectangular kernel.
Local Averaging

Local Averaging

$\bar{Y} | x_0$

$W_i$

kernel = rectangular
bandwidth = FALSE | span = 0.3
Local Averaging: Example

Local Averaging

Occup. Experience
Monthly Net Income
> loc.av <- function(y, x, w)
> {
>     # y ... vector of dependent variable
>     # x ... vector of independent variable
>     # w ... width of the window for averaging w = 2*h
>     x.val <- sort(unique(x))
>     avrg <- function(y, x, x.loc, w) {
>         mean(y[x >= x.loc-w/2 & x <= x.loc+w/2])
>     }
>     m.vec <- sapply(x.val, avrg, y = y, x = x, w = w)
>     cbind(x = x.val, y = m.vec)
> }

> out.av <- with(incex, loc.av(income, oexp, 5))
> plot(income ~ oexp, data = incex, cex = .4)
> lines(out.av, col = 'red', lwd = 3)
Kernel Estimation

We can do a local averaging by using different weighting functions, i.e., kernels.

Weights are defined by kernels (like in kernel density estimation), e.g.

- Rectangular (uniform) kernel: same weight for all observations within the window = local averaging
- Triangular, tricube, Epanechnikov, or normal kernel: observations closer to the $x_0$ under consideration get more weight than observations further away
Kernel Estimation: Normal Kernel

Local Averaging

kernel = normal
bandwidth = FALSE | span = 0.3
Kernel Estimation: Tricube Kernel

Local Averaging

kernel = tricube
bandwidth = FALSE | span = 0.3
Kernel Estimation: Example

Kernel estimation with a normal kernel

Kernel Estimation

Occup. Experience

Monthly Net Income
> k.smth <- function(y, x, h)
  {
    # computes a weighted mean 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 (std. dev. of the normal kernel

    x.val <- sort(unique(x))
    avrg <- function(y, x, x.loc, h) {
      wt <- dnorm((x - x.loc) / h)
      (y %*% wt) / sum(wt) # or: weighted.mean(y, wt)
    }
    m.vec <- sapply(x.val, avrg, y = y, x = x, h = h)
    cbind(x = x.val, y = m.vec)
  }
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 $w_i = K[(x_i - x_0) / h]$

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

where the estimated mean value at $x_0$ is given by the estimated intercept $A$: $\bar{Y} \mid x_0 = A$
Local Linear Regression (with a normal kernel)

kernel = normal
bandwidth = FALSE | span = 0.3
Local Quadratic Regression
(with an Epanechnikov kernel)

Local Quadratic Regression

kernel = epanechnikov
bandwidth = FALSE | span = 0.3
Local Polynomial Regression: Example

Local Polynomial Regression

Occup. Experience

Monthly Net Income
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

R-function: \texttt{loess()}

\texttt{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)
> # 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 == 'm', span = .5, degree = 2)
> out.f <- loess(income ~ oexp, incex, subset = sex == 'f', 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)
Local/Locally Weighted Regression (LOESS/LOWESS)
Local/Locally Weighted Regression
(LOESS/LOWESS)

Dependence on span

loess()

span = .75
span = .3
span = .1
Comparison of Regression Smoothers

Regression Smoothers

- local averaging
- kernel estimation
- local linear regression
- local quadratic regression
- loess (degree = 1)
- loess (degree = 2)
Regression Smoothers: Comparison

Regression Smoothers

- local averaging
- kernel estimation
- local linear regression
- local quadratic regression
- loess (degree = 1)
- loess (degree = 2)
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

\[ \bar{Y} | x = A + Bx \]

\( Bx \)

\( A \)

Occup. Experience

Monthly Net Income