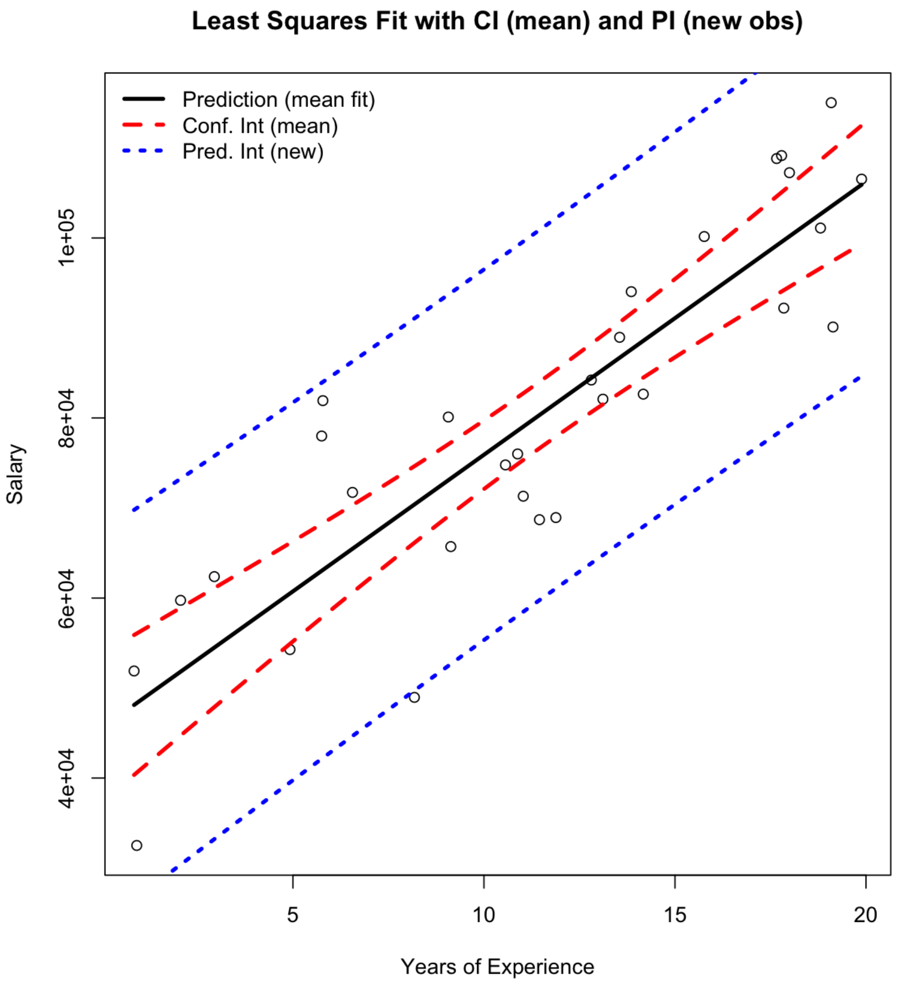
LECTURE 15: CURVE FITTING III

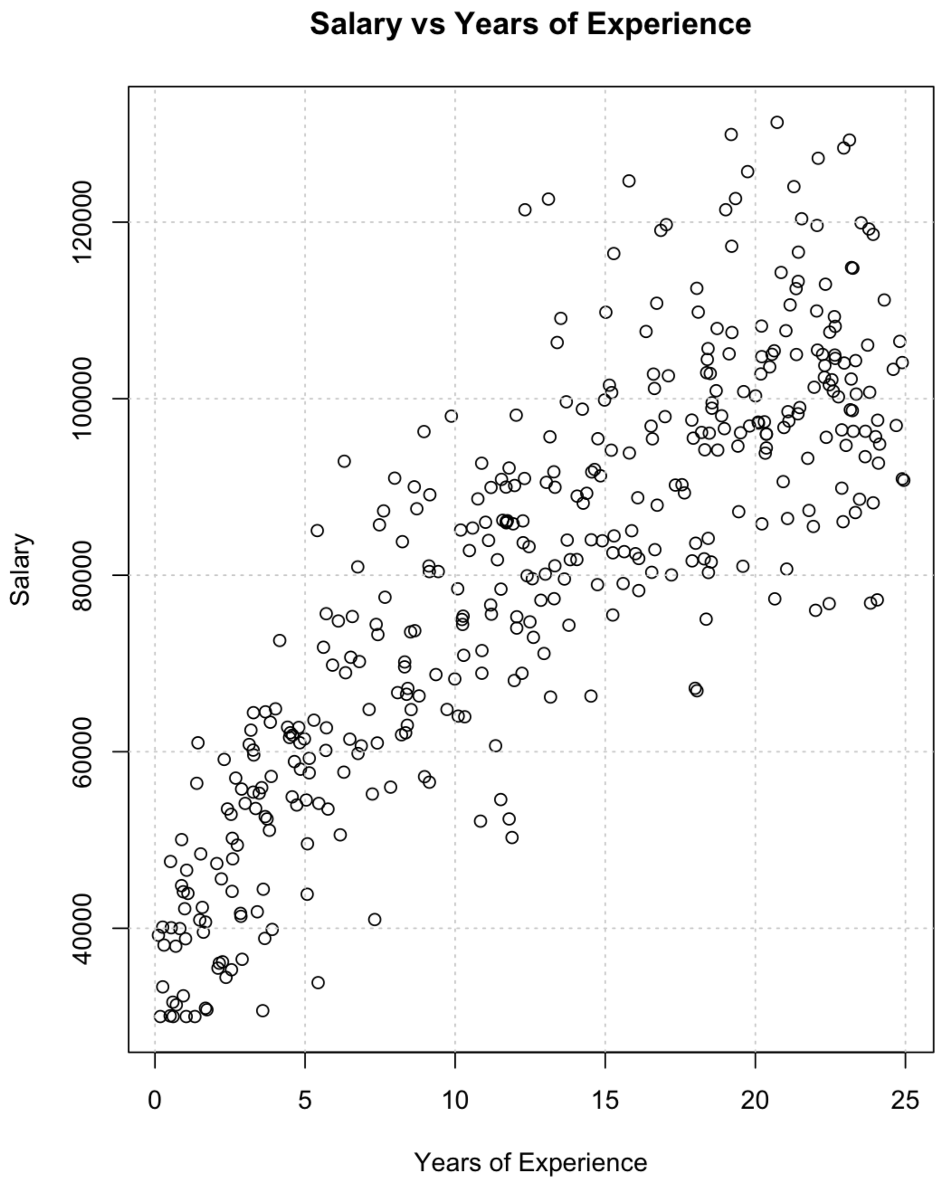
POLYNOMIAL REGRESSION
LOCAL FITTING

OCT 16 2025

INTERVAL "LINES"

- For any specific x_0, the fitted (mean) value is:
  - $\hat{y}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_0$
- The standard error of that mean is:
  - $$\mathrm{SE}_{\hat{y}0} = \sqrt{\sigma^2\left(\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum (x_i - \bar{x})^2}\right)}$$
- The 95% confidence interval for the mean response is:
  - **$\hat{y}_0 \pm t^* \times SE(\hat{y}_0)$**
- The lines for these intervals are **not parallel**
  - They widen away from $\bar{x}$ because $SE(\hat{y}_0)$ increases as x_0 moves away from the mean.
  - More a **"bowtie" shape**
- And the prediction interval lines are bowties as well
  - **$\hat{y}_0 \pm t^* \times SE(pred(x_0))$**



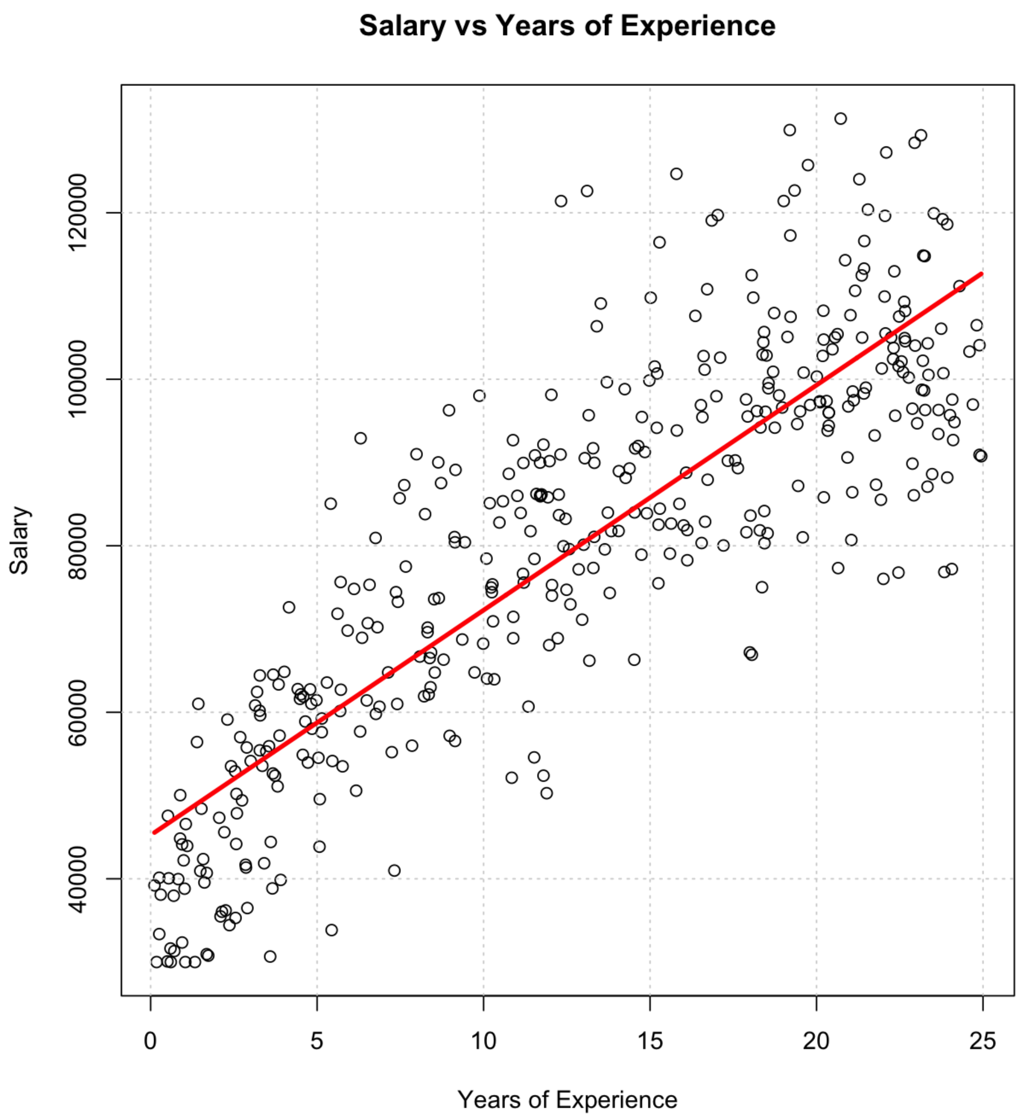

Salary vs Years of Experience
Salary
Years of Experience
40000
60000
80000
100000
120000
0
5
10
15
20
25

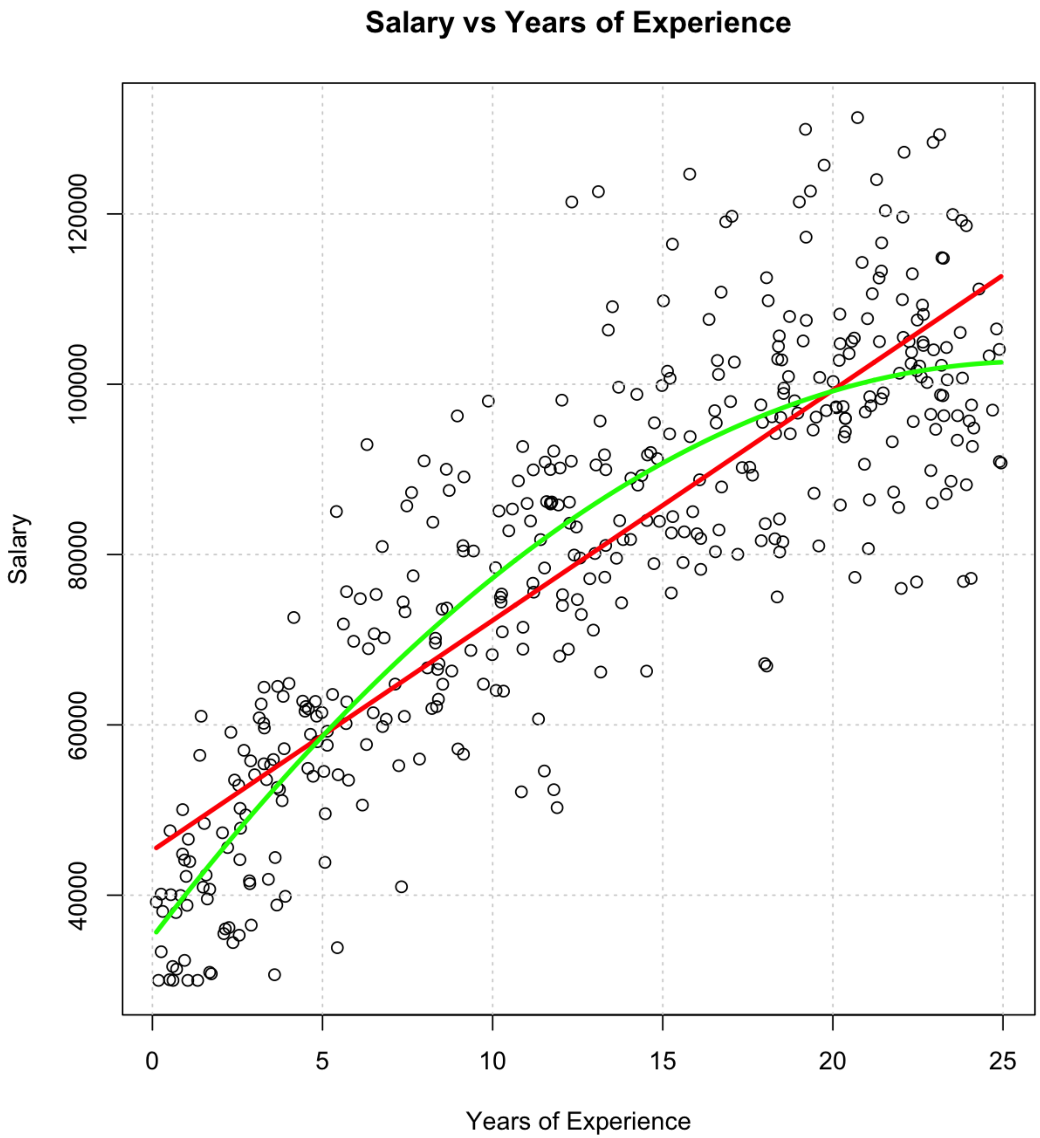
POLYNOMIAL REGRESSION

- We could a quadratic function:
  - $y = \beta_0 + \beta_1 x + \beta_2 x^2 + e$
- Determine optimal coefficents:
  - $\hat{y}_i(\beta_0, \beta_1, \beta_2) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2,$
- Error:
  - $\ell(y_i, \hat{y}_i(\beta_0, \beta_1, \beta_2))$
- Apply least squares
  - $\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i - \beta_2 x_i^2)^2$

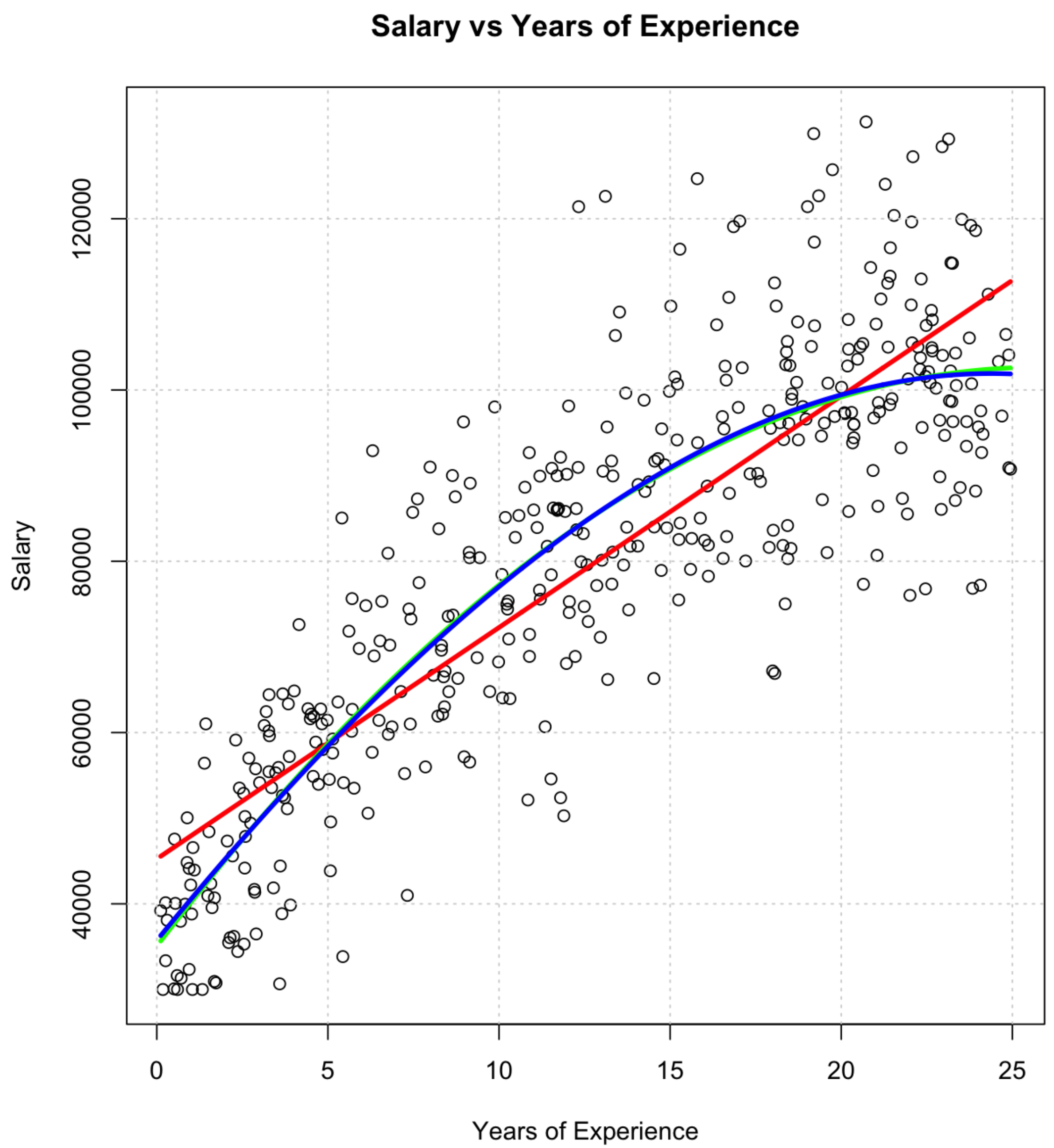
SOLVING FOR POLYNOMIALS

- Polynomial regression is still *linear in parameters*: we just include X^2, X^3, … as extra columns in X.
- The goal is the same: find coefficients β_0, β_1, β_2, … that minimize the sum of squared errors.
- Matrix form: $Y = X\beta + \varepsilon$ where $X = [1, X, X^2, \ldots]$.
- Least-squares solution: $\hat{\beta} = (X^T X)^{-1} X^T Y$.
- So polynomial regression simply solves a **larger linear system** using the same principle as simple linear regression.


Salary vs Years of Experience
Salary
Years of Experience
40000
60000
80000
100000
120000
0
5
10
15
20
25



Salary vs Years of Experience
Salary
120000
100000
80000
60000
40000
0
5
10
15
20
25
Years of Experience


Salary vs Years of Experience
Salary
120000
100000
80000
60000
40000
0
5
10
15
20
25
Years of Experience

INTERPRETATION OF POLYNOMIAL TERMS

- β_1, β_2,… describe curvature, not direct linear effects
- Each higher-order term refines fit near extremes of X
- Interpretation focuses on overall curve shape, not single coefficient
- Higher degrees increase flexibility but risk overfitting
- Choose degree using residual plots or cross-validation
- Underfitting: trend missed; Overfitting: noise captured

POLYNOMIAL REGRESSION

- In polynomial regression, the design matrix X has columns $[1, x, x^2, x^3, \ldots]$
- **Model setup**
  - Polynomial regression model: $Y = X\beta + \varepsilon$, where $X = [1, X, X^2, \ldots, X^d]$.
    - The goal is to find the coefficients β that minimize total squared error.
- **Define the loss function**
  - We measure total squared residuals using $L(\beta) = (Y - X\beta)^T (Y - X\beta)$.
    - This is like adding up all squared differences between the observed and predicted values.
- **Expand the expression**
  - When expanded, the loss becomes $L = Y^T Y - 2\beta^T X^T Y + \beta^T X^T X\beta$.
    - Each term reflects a different interaction between data, model, and parameters

SOLVING

- **Differentiate the loss**
  - Using matrix calculus rules:
    - If $L = b^T\beta$, then its derivative is b
    - If $L = \beta^T A\beta$ where A is symmetric, the derivative is $2A\beta$
- **Simplify and solve**
  - The derivative of the loss is $\partial L/\partial\beta = -2X^T Y + 2X^T X\beta$
    - Setting this equal to zero gives the minimum: $X^T X\beta = X^T Y$
  - Solving for β gives $\hat{\beta} = (X^T X)^{-1} X^T Y$
- **Key idea**
  - Even though the predictors include X, X^2, X^3 and so on, the model is still linear in β
  - So the least-squares solution works exactly like in simple linear regression, only in higher dimensions

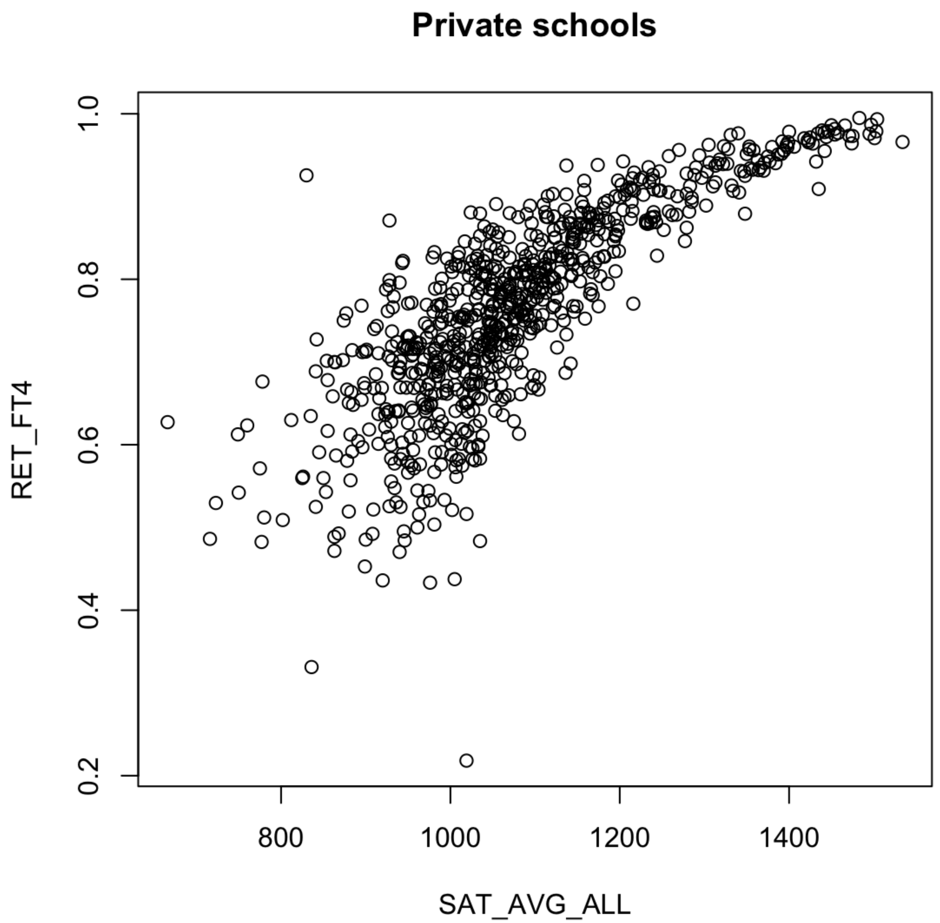
LINEAR ALGEBRA

- **Transpose (X^T)**
  - The transpose flips rows and columns so matrix multiplication can work correctly.
  - Example: `X = [[1, 2, 3], [4, 5, 6]]` becomes `Xᵀ = [[1, 4], [2, 5], [3, 6]]`
  - Like rotating the data table so that all variables align for dot products.
- **Inverse (A^{-1})**
  - The inverse is the matrix version of dividing by a number.
  - For numbers, dividing by 4 is the same as multiplying by ¼.
  - For matrices, multiplying by A^{-1} undoes the effect of multiplying by A, since $A^{-1}A = I$.
  - In regression, $(X^TX)^{-1}$ removes overlap among predictors, so each coefficient β is adjusted for the others.
    It is like dividing out the entanglement among predictors.
- **Pseudoinverse (X^+)**
  - Defined as $(X^TX)^{-1}X^T$ when X^TX can be inverted.
  - This is the matrix equivalent of "smart division" by X, used when X is not square or directly invertible.
  - It gives the compact solution $\hat{\beta} = X^+Y$

LOCAL FITTING

FUNCTIONS

- What would be a good fit here ?

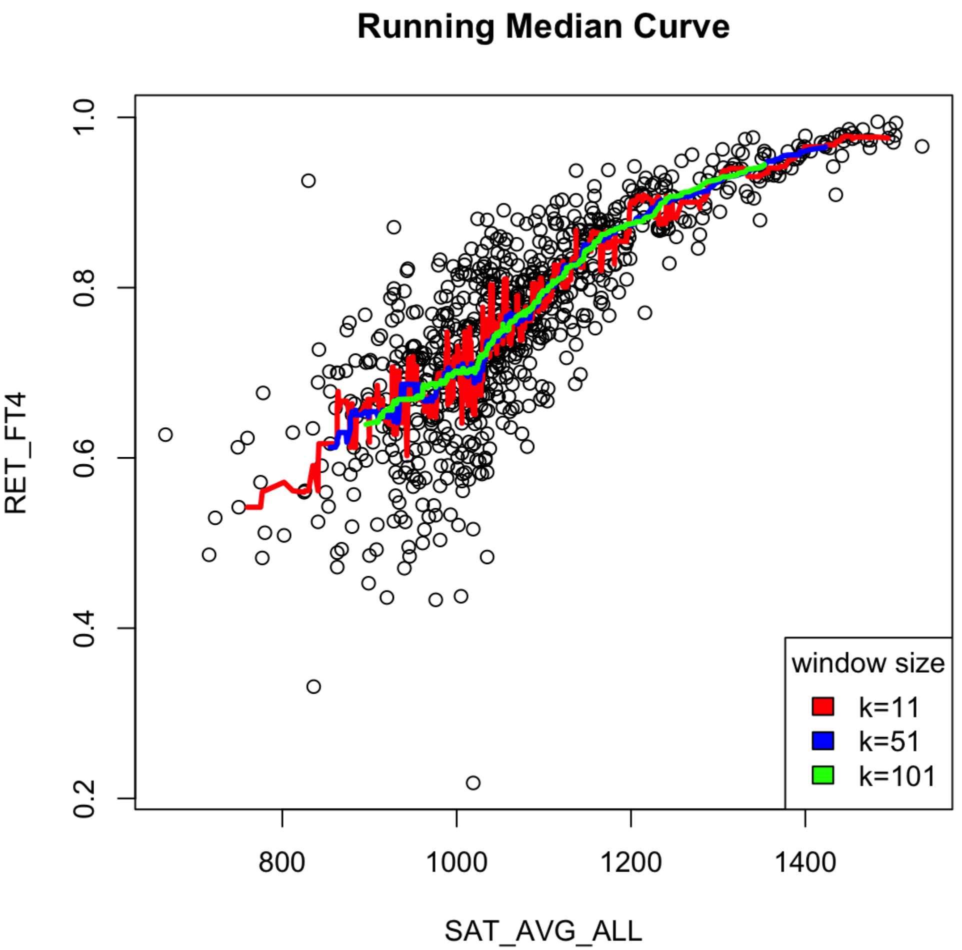


LOCAL FITTING OVERVIEW

- We can have a **single complex** function
  - Or, combine **multiple simple** functions
- Local fitting methods estimate the relationship between X and Y using **nearby data points**
  - Rather than a single global model
- Each fitted value is based on a small neighborhood around x_0.
- This produces flexible, smooth curves that adapt to local structure in the data

SLIDING WINDOWS
RUN THE MEAN (OR MEDIAN)

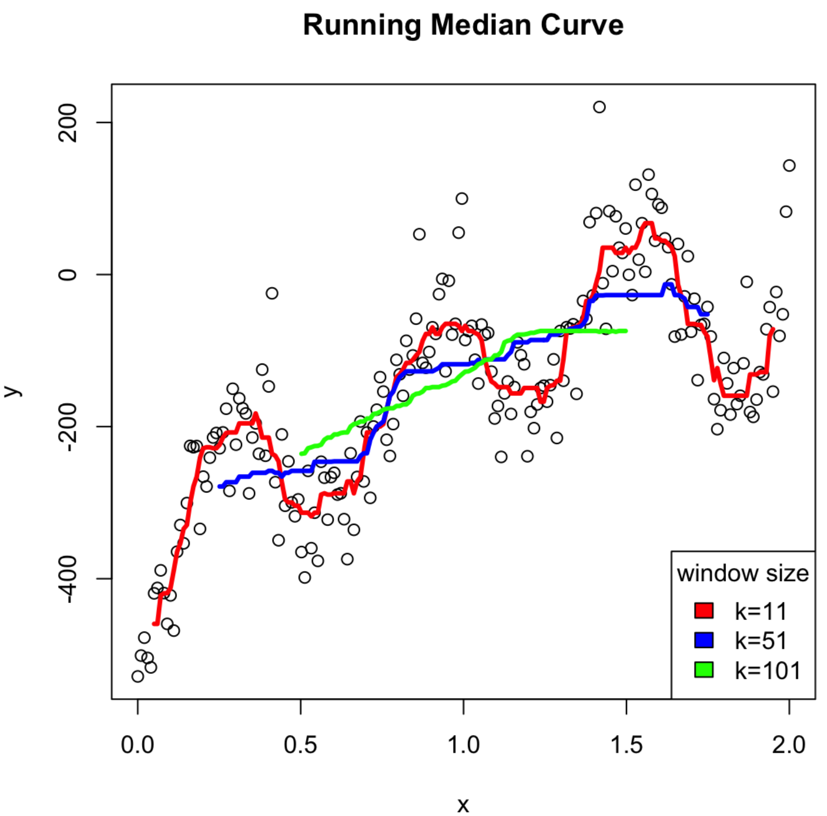
- $$\hat{f}(x) = \frac{1}{\text{\#in window}} \sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} y_i$$



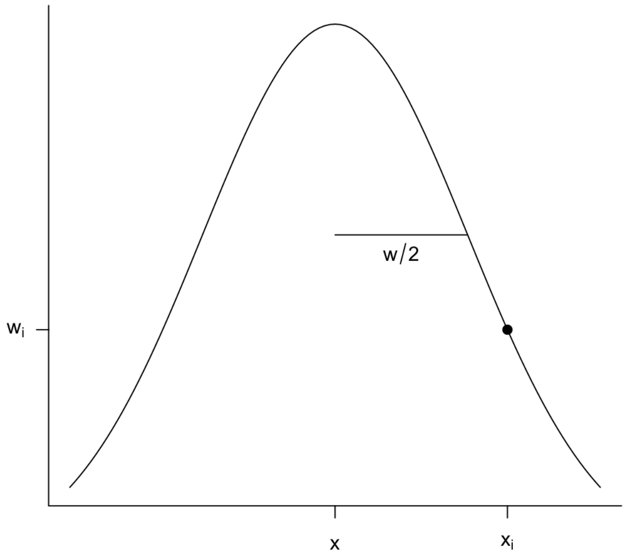
KERNEL WEIGHTING

$$\hat{f}(x) = \frac{\sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} y_i}{\sum_{i:x_i \in [x-\frac{w}{2}, x+\frac{w}{2})} 1}$$

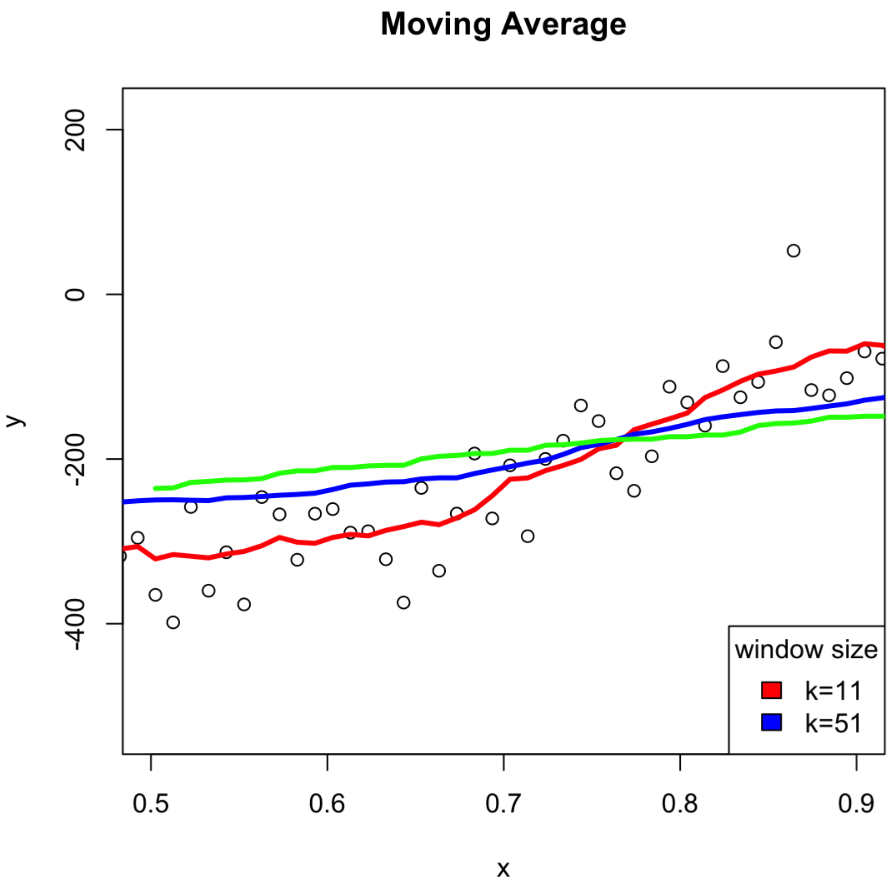
- $$= \frac{\sum_{i=1}^{n} y_i f(x, x_i)}{\sum_{i=1}^{n} f(x, x_i)}$$
- $$f(x, x_i) = \begin{cases} \frac{1}{w} & x_i \in [x - \frac{w}{2}, x + \frac{w}{2}) \\ 0 & otherwise \end{cases}$$
- **Nadarya-Watson Kernel**
- How does it compare with running mean/ median ?
- What is the effect of the window size ?



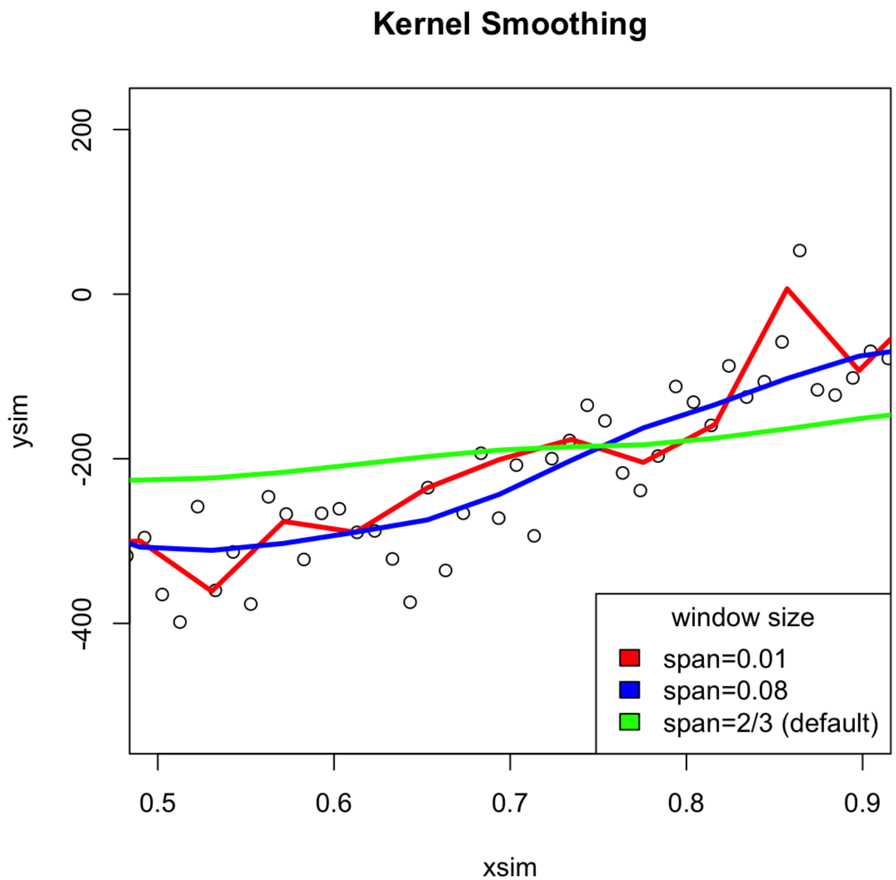
GAUSSIAN KERNEL



- Is a popular choice



Moving Average
y
x
200
0
-200
-400
0.5
0.6
0.7
0.8
0.9
window size
k=11
k=51



Kernel Smoothing
ysim
xsim
200
0
-200
-400
0.5
0.6
0.7
0.8
0.9
window size
span=0.01
span=0.08
span=2/3 (default)

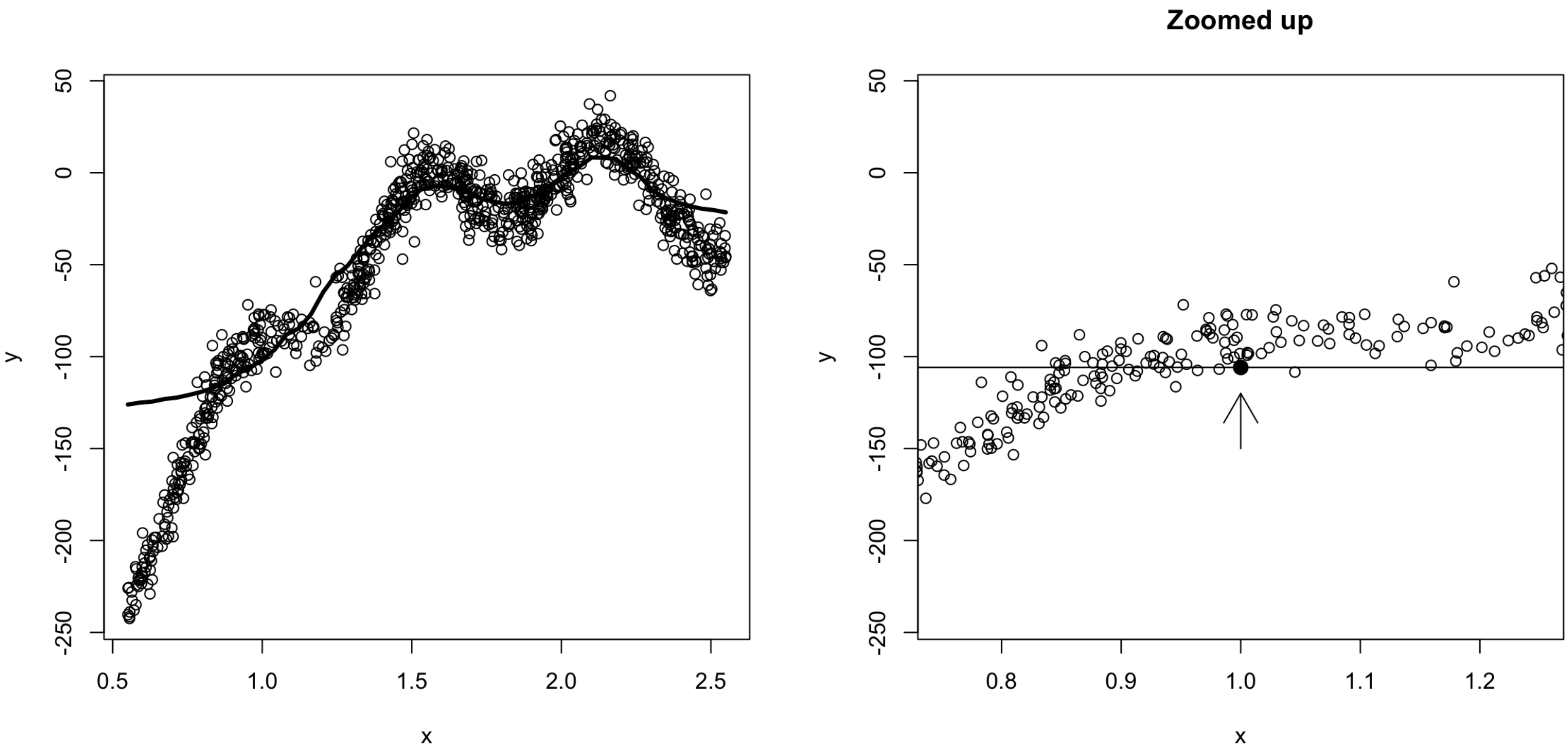
MATHEMATICAL FORMULATION

- At **each** target x_0
  - Estimate coefficients $\beta_0(x_0)$, $\beta_1(x_0)$ by minimizing
  - $\Sigma\ K((x_i - x_0)/h) \times (y_i - \beta_0 - \beta_1 x_i)^2$
- K is a kernel function **assigning higher weight to nearby** x_i
- h (bandwidth) controls how wide the neighborhood is

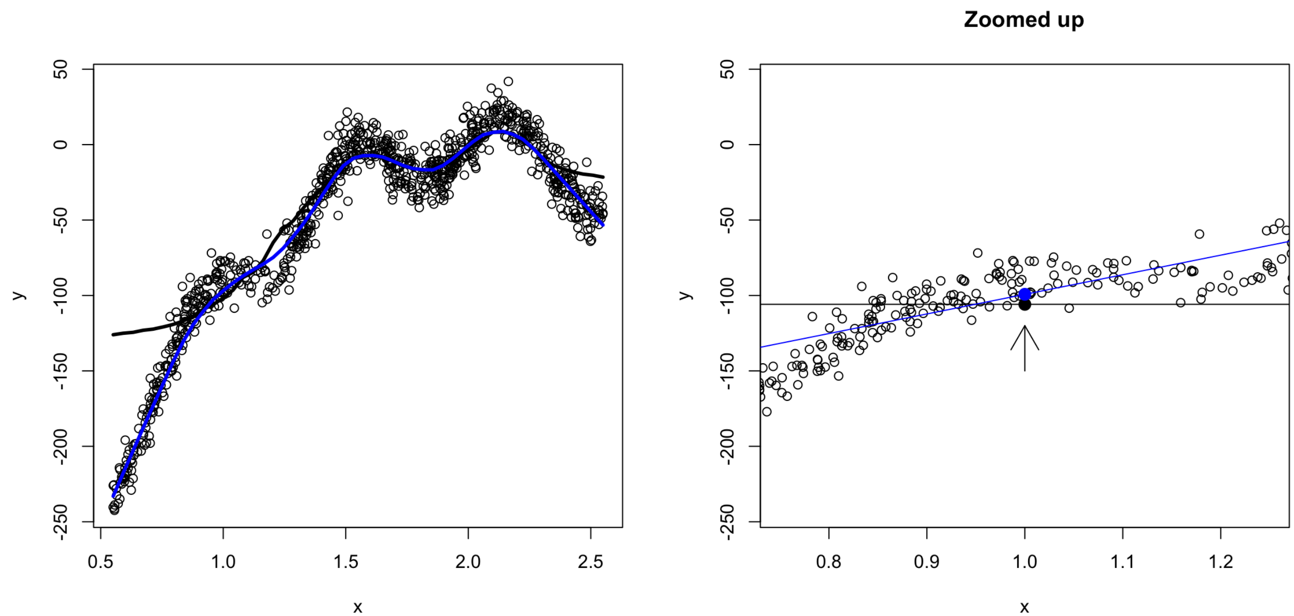
KERNEL WEIGHTING FUNCTIONS

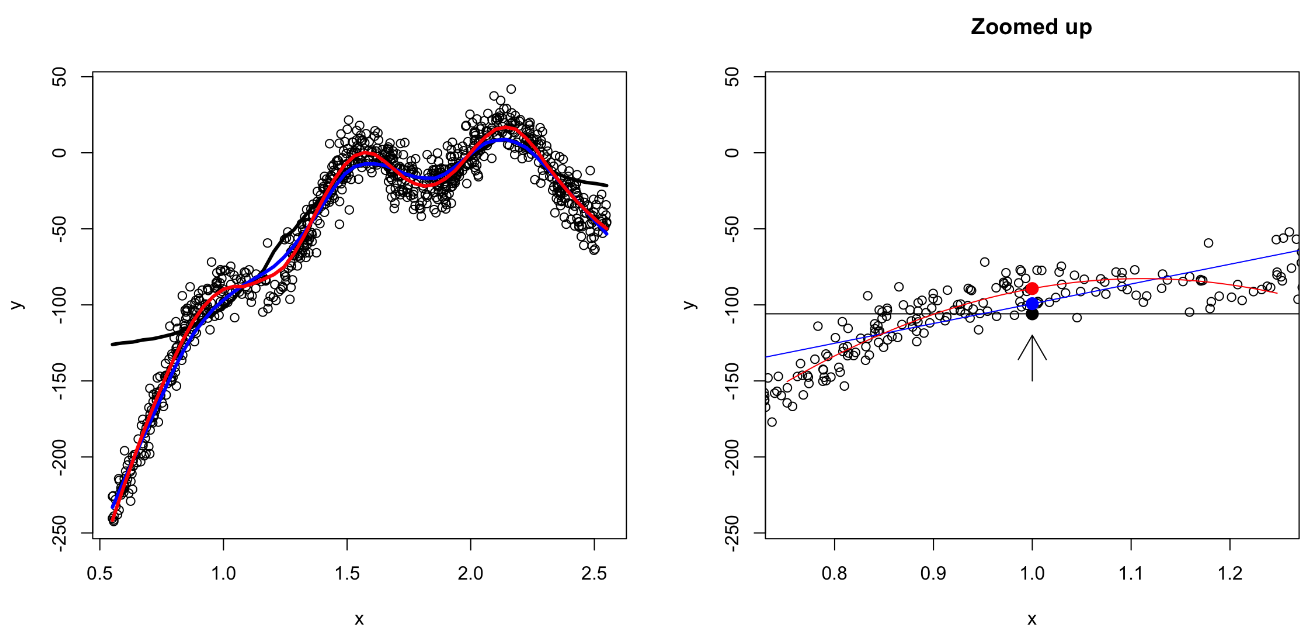
- Kernel weights determine how quickly influence declines with distance from x_0.
- Common choices: Gaussian, Epanechnikov, Tricube kernels
- Larger $|x_i - x_0| \Rightarrow$ smaller weight $K((x_i - x_0)/h)$

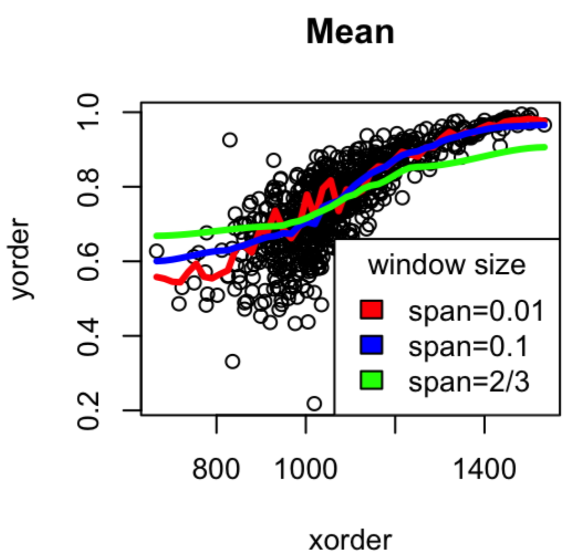
LOESS

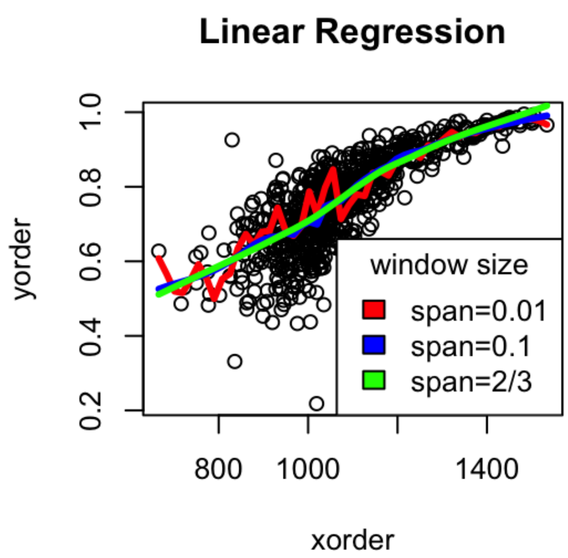


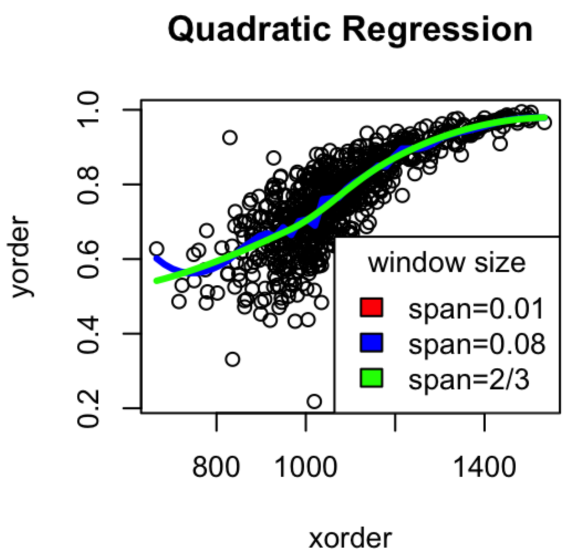
Zoomed up
y
x
50
0
-50
-100
-150
-200
-250
0.5
1.0
1.5
2.0
2.5
0.8
0.9
1.0
1.1
1.2


Zoomed up
y
x
50
0
-50
-100
-150
-200
-250
0.5
1.0
1.5
2.0
2.5
0.8
0.9
1.0
1.1
1.2


Zoomed up
y
x
50
0
-50
-100
-150
-200
-250
0.5
1.0
1.5
2.0
2.5
0.8
0.9
1.0
1.1
1.2


Mean
1.0
0.8
0.6
0.4
0.2
yorder
800
1000
1400
xorder
window size
span=0.01
span=0.1
span=2/3


Linear Regression
1.0
0.8
0.6
0.4
0.2
yorder
800
1000
1400
xorder
window size
span=0.01
span=0.1
span=2/3


Quadratic Regression
1.0
0.8
0.6
0.4
0.2
yorder
800
1000
1400
xorder
window size
span=0.01
span=0.08
span=2/3

LOESS

- LOESS (Locally Estimated Scatterplot Smoothing) fits **many small regressions** instead of one global line
- At each target point x_0 , it fits a **local polynomial** (usually linear or quadratic) using **weighted nearby points**
- Nearby points get **more weight**, distant points get **less weight**
- The fitted value at x_0 is the prediction from that local weighted regression
- It is a **non-parametric** method, we don't assume one global β for the entire dataset

THE MATH

- At each x_0
  - We minimize the **weighted least-squares loss**:
    - $L(\beta \mid x_0) = \Sigma_i\, w_i(x_0)\, [y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i^2 \ldots)]^2$
  - The weight $w_i(x_0)$ is **largest near x_0**
- The **Tricube** kernel

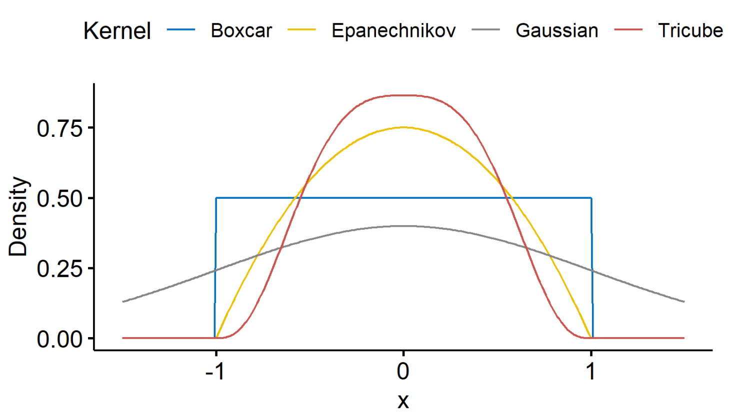
  $w_i(x_0) = (1 - |x_i - x_0|^3 / d^3)^3$ for $|x_i - x_0| < d$, else 0

- The solution at x_0 gives the local coefficients:

  $$\hat{\beta}(x_0) = (X^T W(x_0) X)^{-1} X^T W(x_0) Y$$

- The fitted value:

  $$\hat{y}(x_0) = [1, x_0, x_0^2, \ldots]\, \hat{\beta}(x_0)$$



SOLUTION

- We solve: $\hat{\beta}(x_0) = (X^T W X)^{-1} X^T W Y$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}, \quad W = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & w_3 \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Then:

$$\hat{\beta}(x_0) = (X^T W X)^{-1} X^T W Y$$

This yields local slope and intercept (β_0, β_1) at point x_0.

EXAMPLE

- Suppose we have 3 nearby points around $\mathbf{x}_0$ **= 5**
  - x = [4, 5, 6]
  - y = [10, 15, 18]
- Weights using the tricube kernel):
  - w = [0.25, 1.0, 0.25]
- Then $\mathbf{x}_0$ **= 5** gets the most weight !

$$X = \begin{bmatrix} 1 & 4 \\ 1 & 5 \\ 1 & 6 \end{bmatrix}, \quad W = \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 0.25 \end{bmatrix}$$

$$\hat{\beta} = (X^T W X)^{-1} X^T W Y$$

$$\hat{\beta} \approx \begin{bmatrix} -5.0 \\ 3.9 \end{bmatrix}$$

$$\hat{y}(5) = -5.0 + 3.9 \times 5 = 14.5$$

LOESS SUMMARY

- LOESS repeats a local weighted regression at each x_0
- Each local fit has its own $\beta_0(x_0)$, $\beta_1(x_0)$, etc.
- Nearby points dominate each local line; faraway points barely influence it
- The collection of all local predictions forms a **smooth curve** that adapts to the data
  - The final "curve" is a combination of small/short straight lines (chain links)