Regression, Prediction and Classification

Jacob LaRiviere
Justin M. Rao

Terminology

X: $features$.
y: outcomes

Goal is to model outcomes as a function of features.

Width of X is p

Length of X and y is n

Terminology cont.

X: $features$.
y: outcomes

Feature 1

Obs 1 → $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$

Obs 1 → $\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$

Terminology cont.

X: $features$.
y: outcomes

Feature 1

Obs 1 → $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$

Obs 1 → $\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$

Estimating equations

$$Y = f(X) + \epsilon$$

$$y_i = f(x_i) + \epsilon_i$$

Where ϵ_i is call the error term.

Example: coin flipping

$$Y = f(X) + \epsilon$$

$$flip_i = p + \epsilon_i$$

$$\text{if outcome} = 1 \text{ then } \epsilon_i = 1 - p$$
$$\text{if outcome} = 0 \text{ then } \epsilon_i = -p$$

p is a constant, called the “success rate”

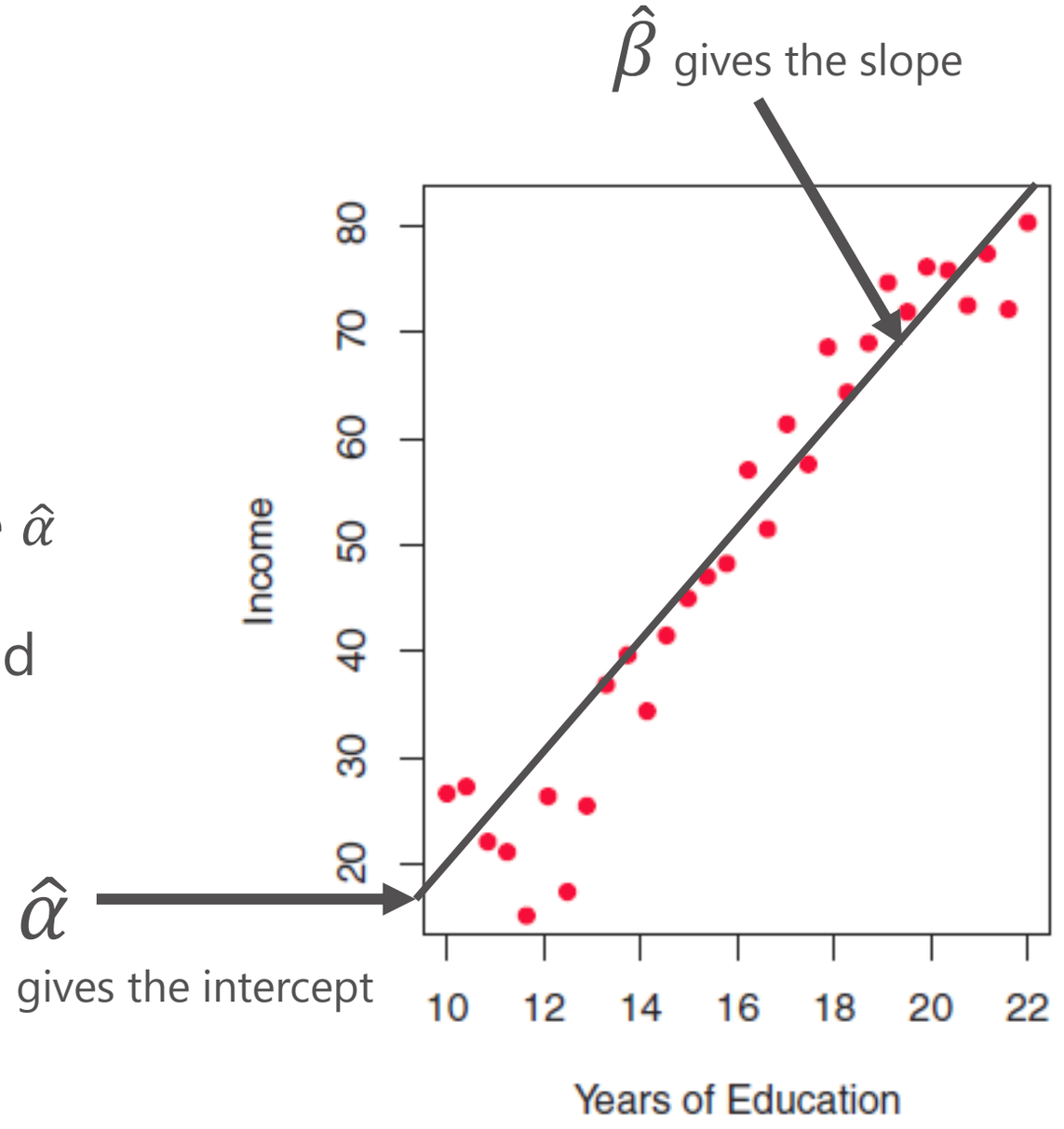
Linear regression

$$y_i = \alpha + \beta_1 x_i + \epsilon_i$$

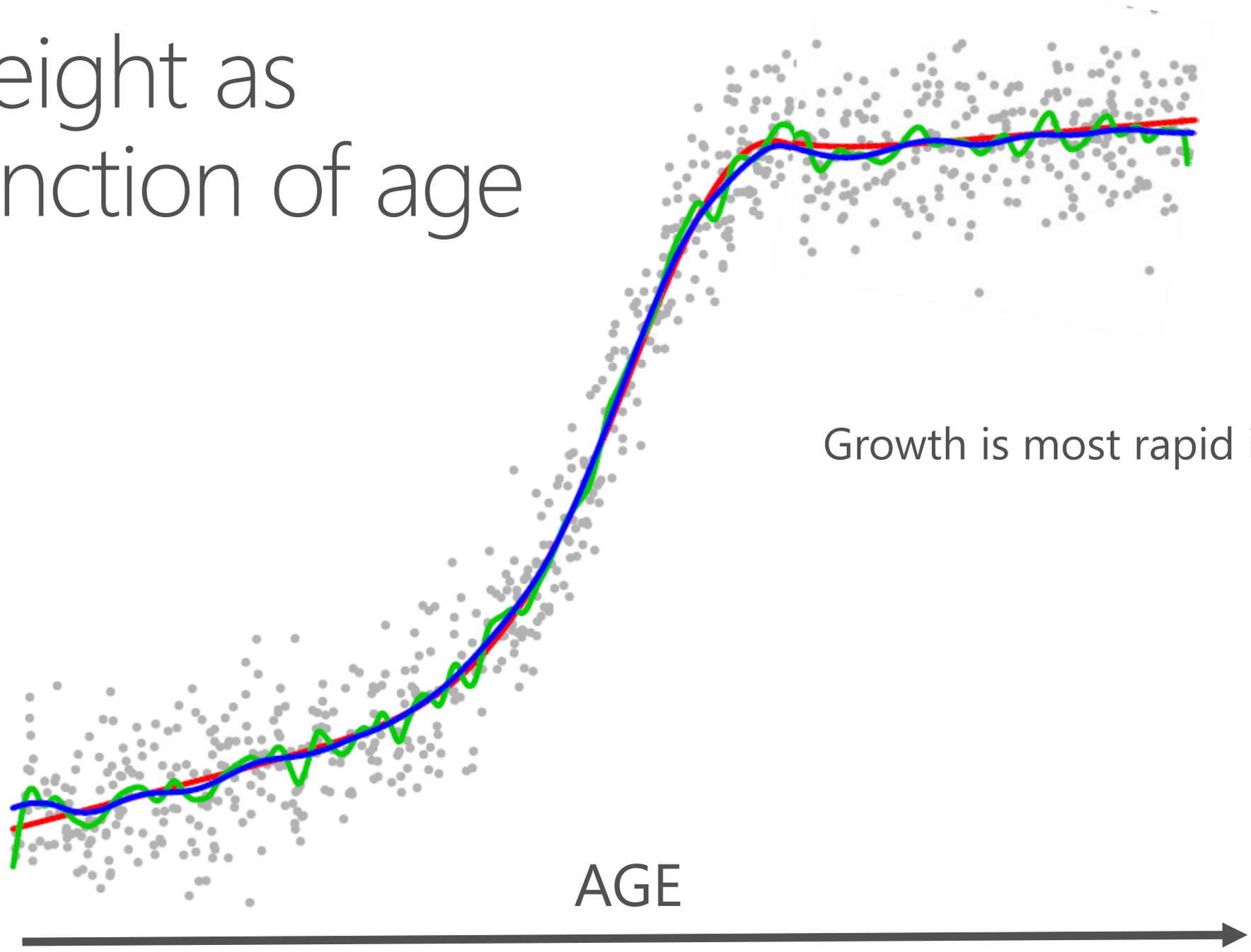
Ordinary least squares will find the $\hat{\alpha}$ and $\hat{\beta}$ to minimized the squared distance between the fitted line and the observations

$$\hat{y_i} = \hat{\alpha} + \hat{\beta} x_i$$

Minimizes $(\hat{y} - y)^2$



Height as function of age



Growth is most rapid in ages 12-16

Height as function of age

If we estimated $h = \alpha + \beta * age$, the constant growth rate β would overstate growth in early and later years, and understate during puberty

How would we estimate this relationship?

General formula:

$$h_i = f(age_i) + \epsilon_i$$

ϵ_i allows people of he same age to have different heights. Even if we have the correct growth model using age alone, we expect some variation in height conditional on age.

To use a linear model, polynomial features allow for a non-linear relationship between the feature, age, and the outcome, height.

$$h_i = \alpha + \beta_1 age_i + \beta_2 age_i^2 + \cdots + \beta_p age_i^p + \epsilon_i$$

Linear regression

Linear regression allows you to use non-linear functions of the features, provided they enter the function as an additive term, with weight β

$$y_i = \alpha + \beta_1 g_1(x_{i,1}) + \cdots + \beta_p g_p(x_{i,p}) + \epsilon_i$$

The simplest example is where all features simply enter without any transformations

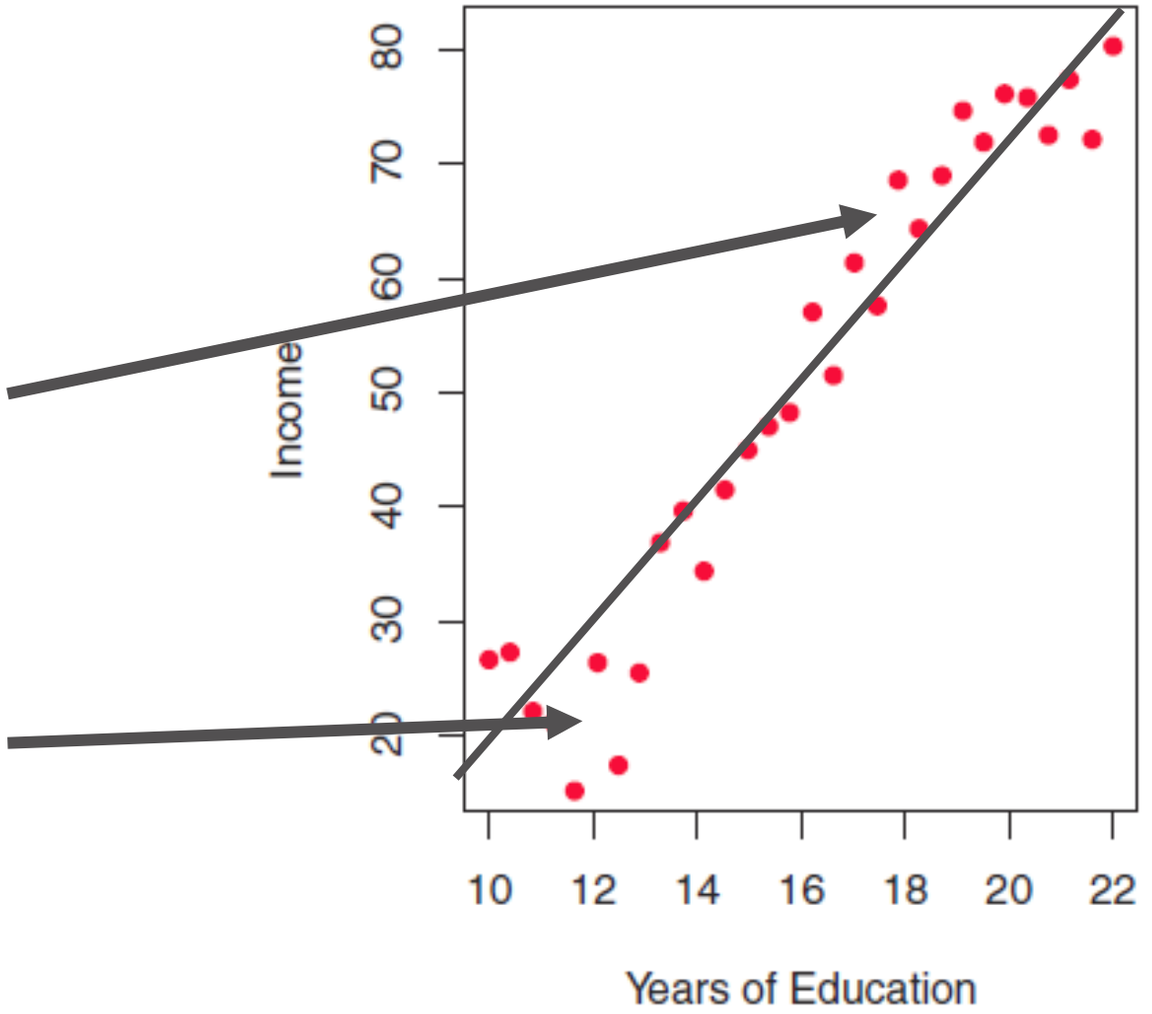
$$y_i = \alpha + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \epsilon_i$$

Linear regression

$$y_i = \alpha + \beta_1 x_i + \epsilon_i$$

A linear function tends to underestimate for medium-high education

And overestimate income for low education

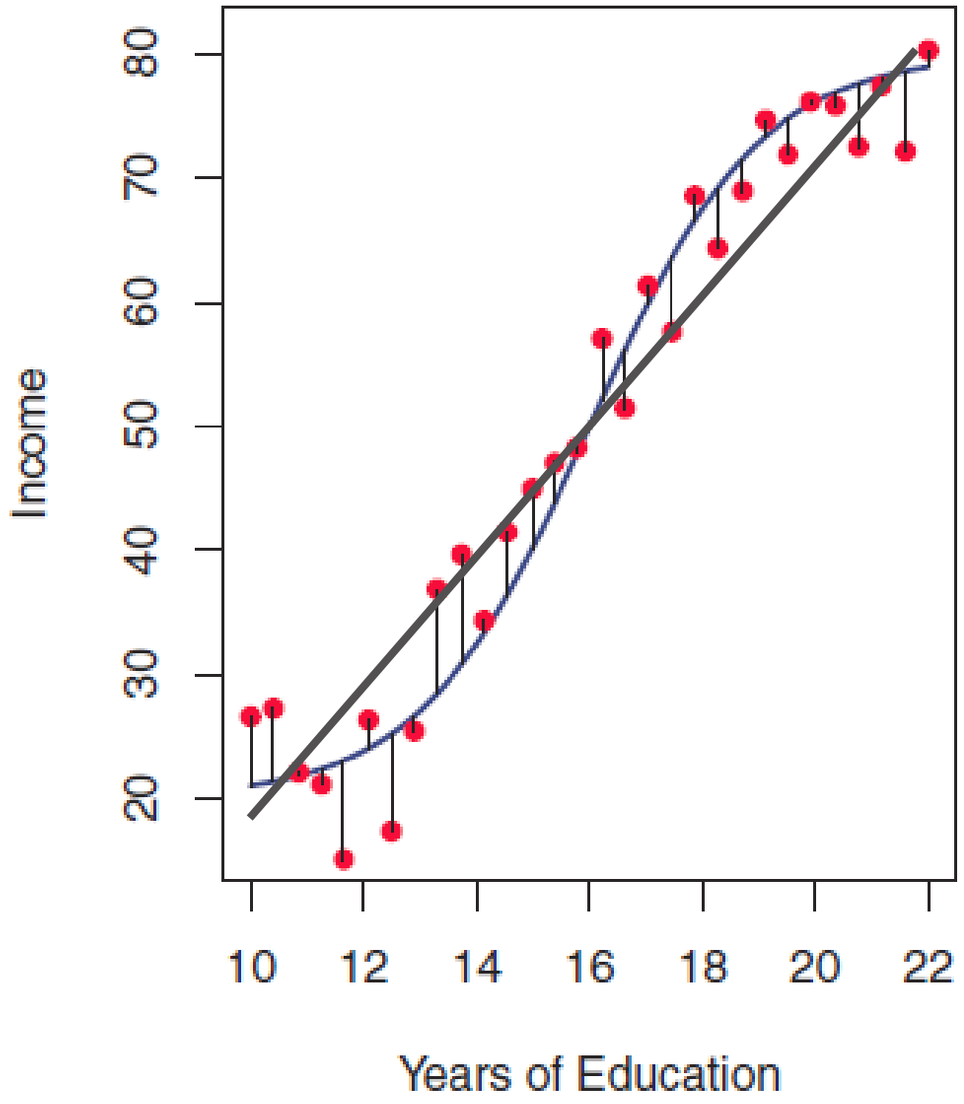


Linear regression

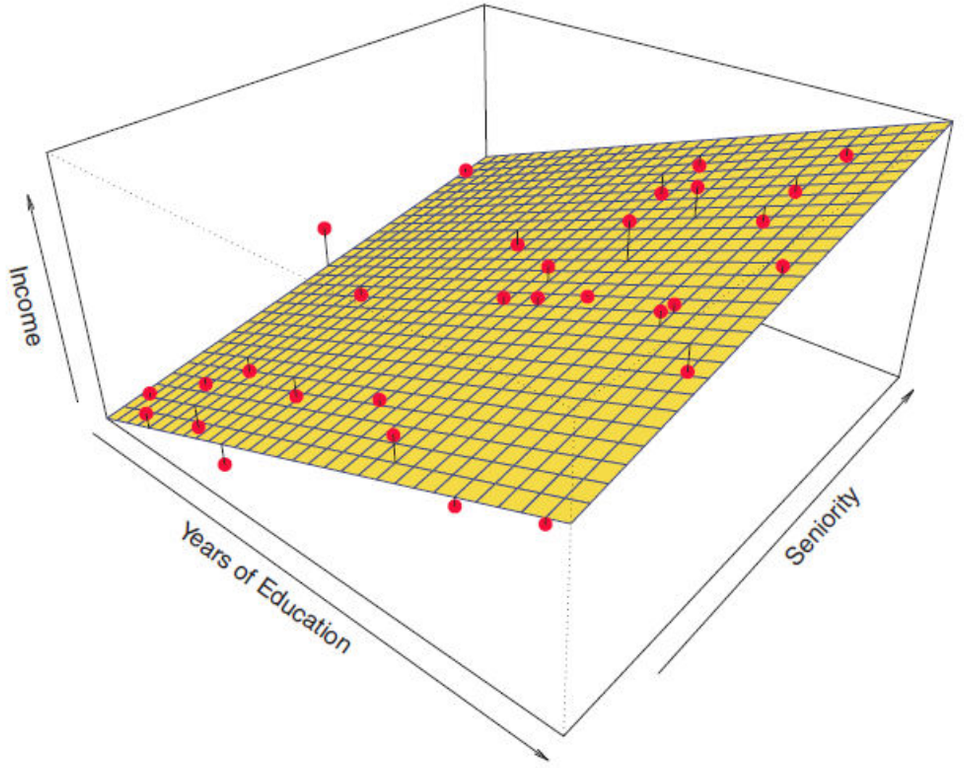
A cubic polynomial fits the data much better

$$\widehat{y_i} = \hat{\alpha} + \widehat{\beta_1} x_i + \hat{\beta}_2 x_i^2 + \hat{\beta}_3 x_i^3$$

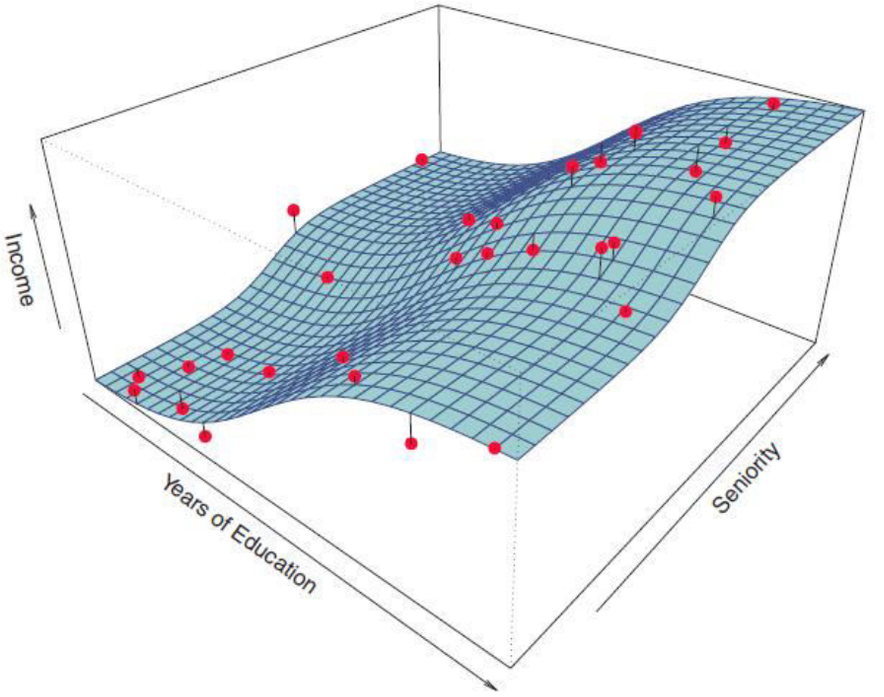
$(\hat{y} - y)^2$ is now much smaller



Visualizing in two dimensions



$$y_i = \alpha + \beta_1 educ_i + \beta_2 seniority + \epsilon_i$$



Higher order polynomial in educ and seniority could approximate this function

Steps for linear regression

1. Define the relationship we are trying to model. What is the outcome? What raw features do we have? E.g.

$$h_i = f(age_i) + \epsilon_i$$

2. Define a linear model to approximate this relationship, e.g.

$$h_i = \alpha + \beta_1 age_i + \beta_2 age_i^2 + \cdots + \beta_p age_i^p + \epsilon_i$$

3. Create the necessary features `eg. Age2=age^2 (or use poly)`
4. **Estimate the model:**

```
mymodel=lm(y~age+age2…+ageP)
summary(mymodel)
```

5. Evaluate the model with an evaluation metric.
6. Repeat steps 2-5 to improve model fit.

Model prediction

1. After we have fit a model, we can predict the outcome for any input.

$$\widehat{h_i} = \alpha + \widehat{\beta_1} age_i + \widehat{\beta_2} age_i^2 + \cdots + \widehat{\beta_p} age_i^p$$

2. For any given observation $\widehat{h_i} - h_i = prediction\ error$

$$(\widehat{h_i} - h_i)^2 : \text{squared loss}$$
$$|\widehat{h_i} - h_i| : \text{absolute loss}$$

Model Fit

Fit refers to how well our model, an approximation of reality, matches what we actually observe in the data

Mean squared error is the average of squared loss for each data point (row) we evaluate

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Training vs. Test

We typically want to “train” the model on the data “we have” and test the model on “new data”

This is a realistic test of how well the model predicts outcomes.

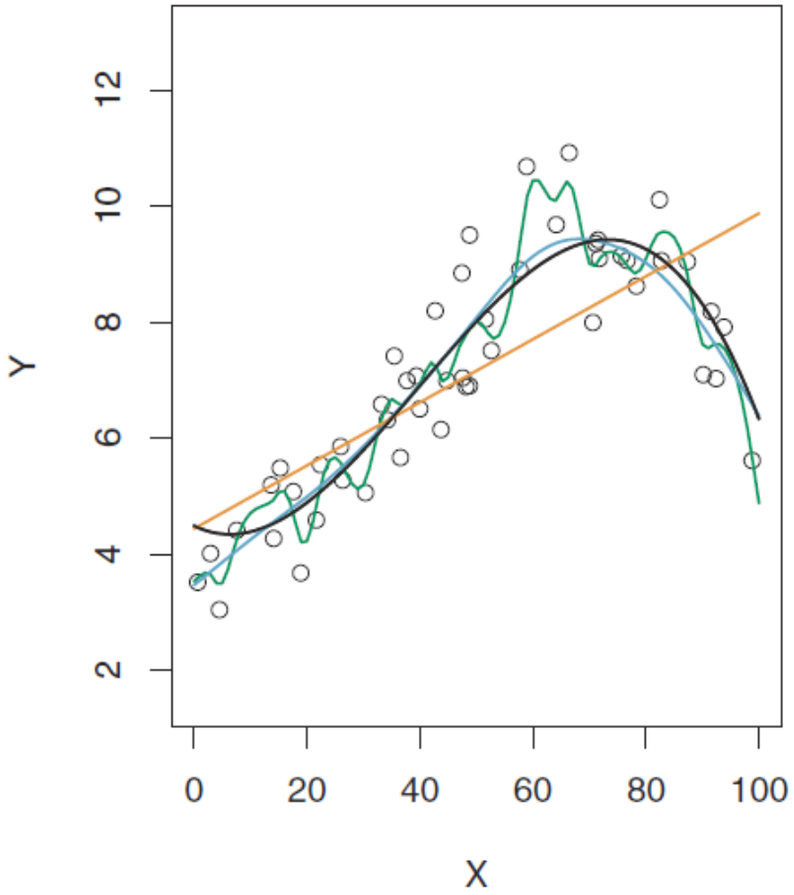
In practice, we will subset our data:

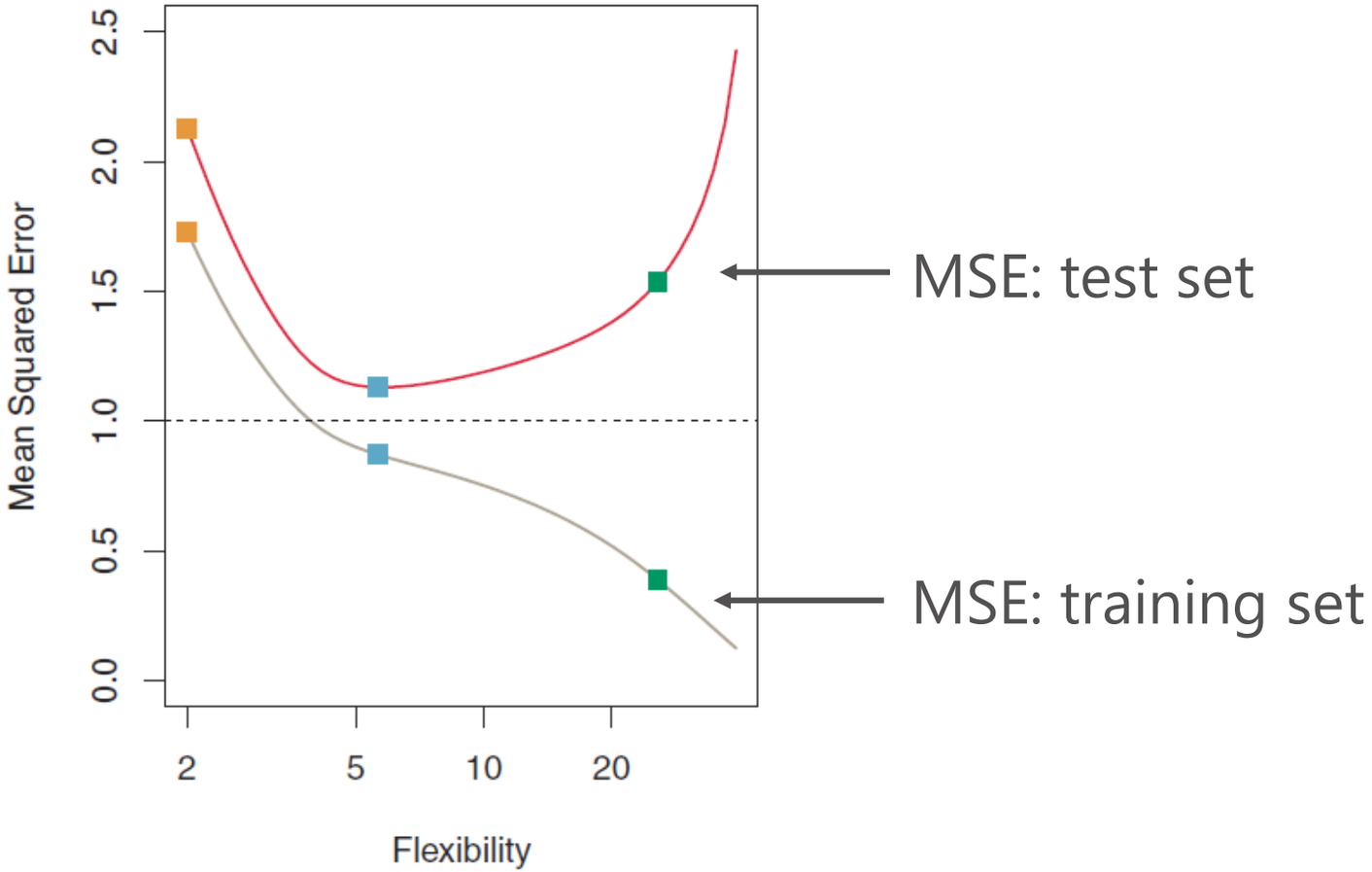
X%: Training

Y%: Test

80% is commonly used for training.

Why do we need training vs. test?



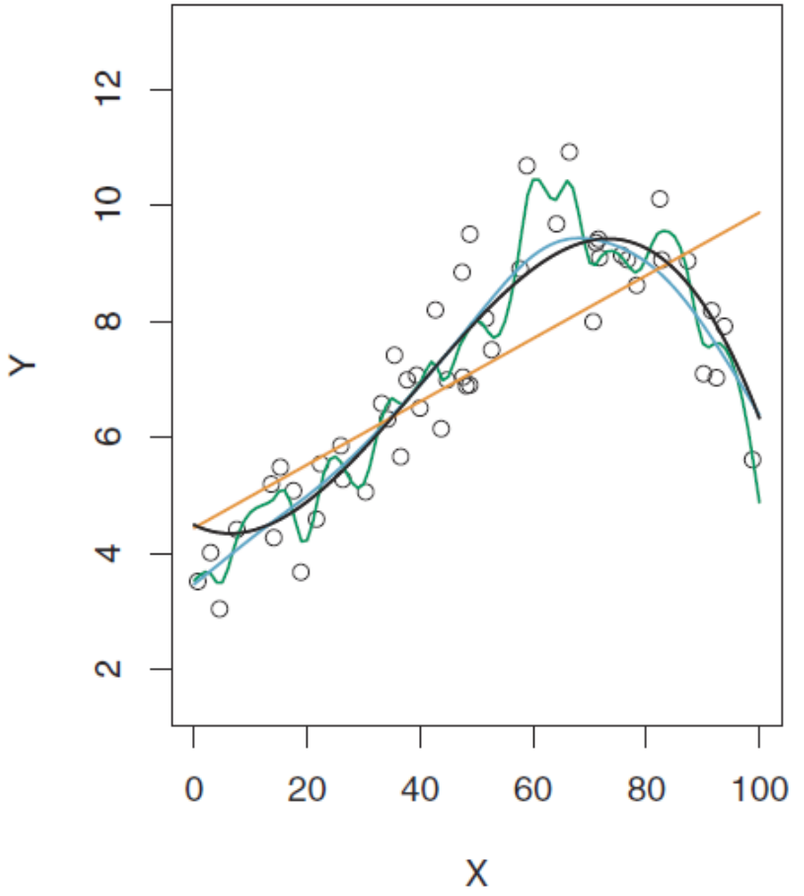


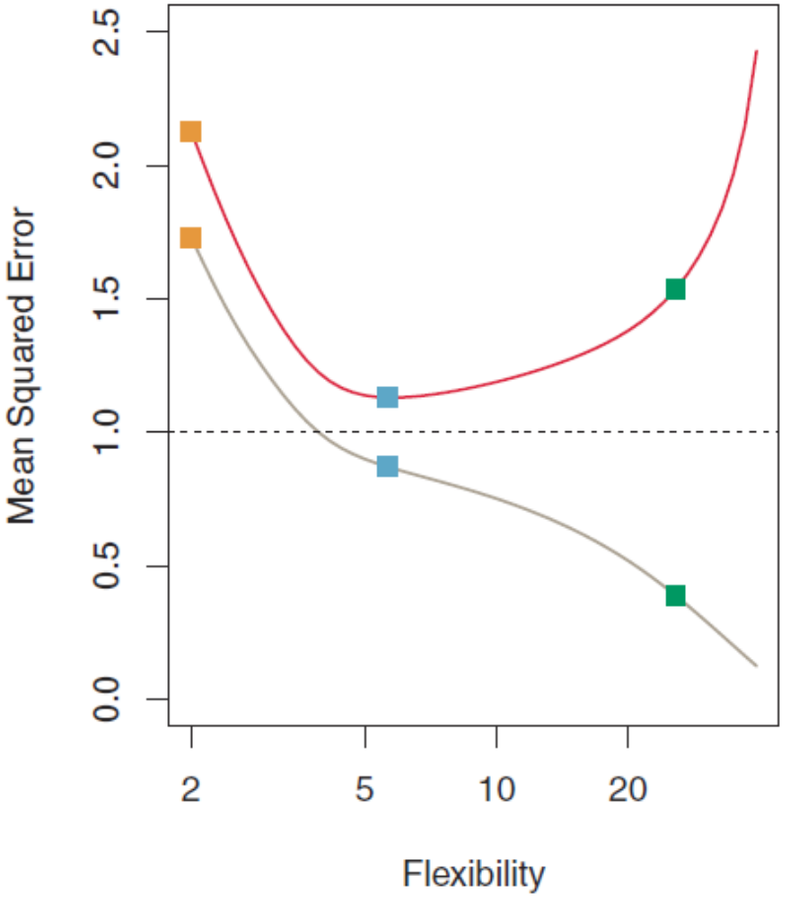
Why do we need training vs. test?

The green curve "overfits" the data. It fits the noise.

Using a "out of sample" test set guards against "overfitting"

The blue curve gets closest to the black (truth) curve





Training vs. test in practice

- Randomly select observations (rows) to be in the test set. The remainder are the training set.
- Why random: want complete coverage. E.g. don't want to train in February and test on July.
- Subset on the training set for all model estimation. E.g. any `data.frame` you pass to `lm`
- Use the `predict` command to make predictions ($\hat{y}$) for the test set
- Compute the *evaluation metric* using test set (you have the observe value and the predicted or modeled value)

R-squared

$$\mathrm{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Residual sum of squares

$$\mathrm{TSS} = \sum(y_i - \bar{y})^2$$

Total variation in outcome variable

$$R^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

Fraction of the variation in the outcome variable captured by the model

A Fair R-squared Measure

- R-squared is reported for the training data. That is, it is "in sample"
- We learned that we typically want to use out-of-sample evaluation metrics, what do you we do
- It turns out that $R^2 = corr(\hat{y}_i, y_i)^2$
- This means, $corr(\hat{y}_i, y_i)^2$, for the test set can be interpreted just like R-squared (% of variation explain) and is a "fair test"

Feature types

Binary features: equal to 1 or 0. For example:

$$x_i = \begin{cases} 1 & \text{if } i\text{th person is female} \\ 0 & \text{if } i\text{th person is male,} \end{cases}$$

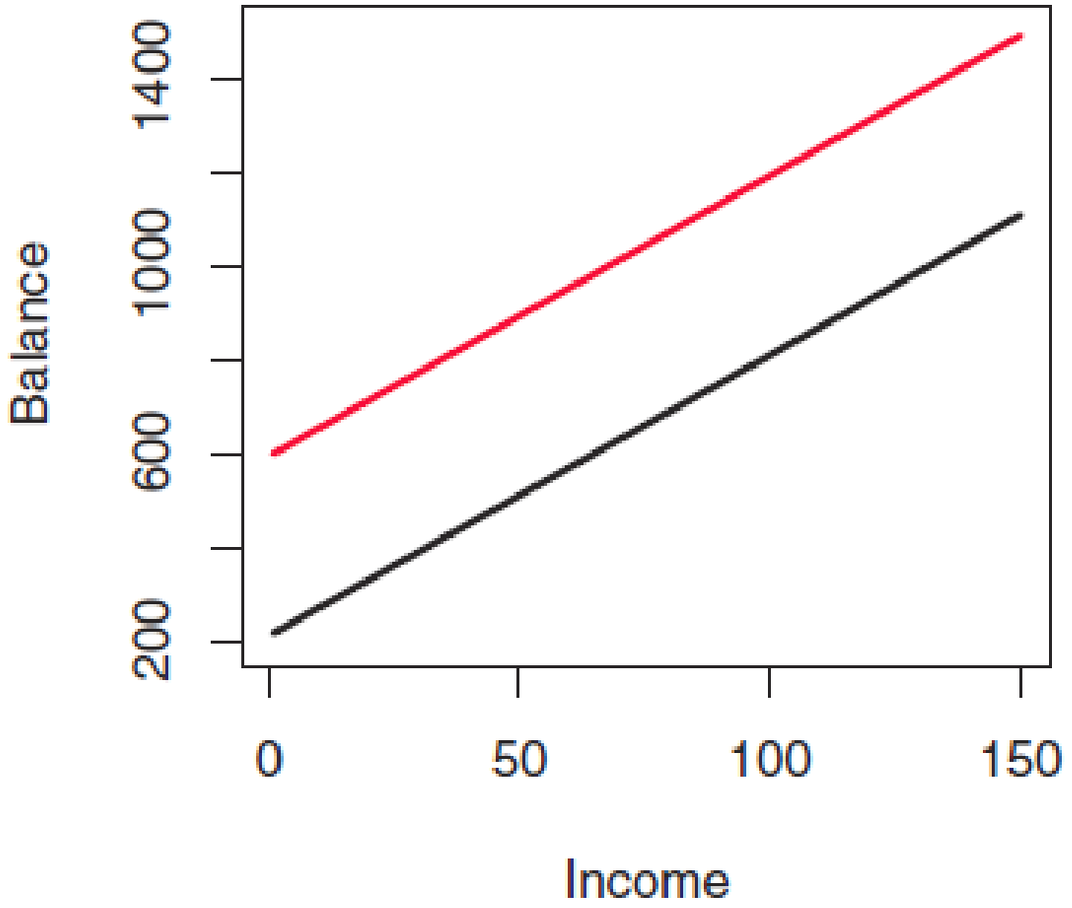
Continuous features: can take an real value

Categorical features: can take one of N values. E.g. Saturday, Sunday, Monday… declaring `as.factor` then R will treat it as N binary variables for each category (ex. =1 if Saturday, =0 otherwise)

Understand binary features

β_1 allows for a different y-intercept for students

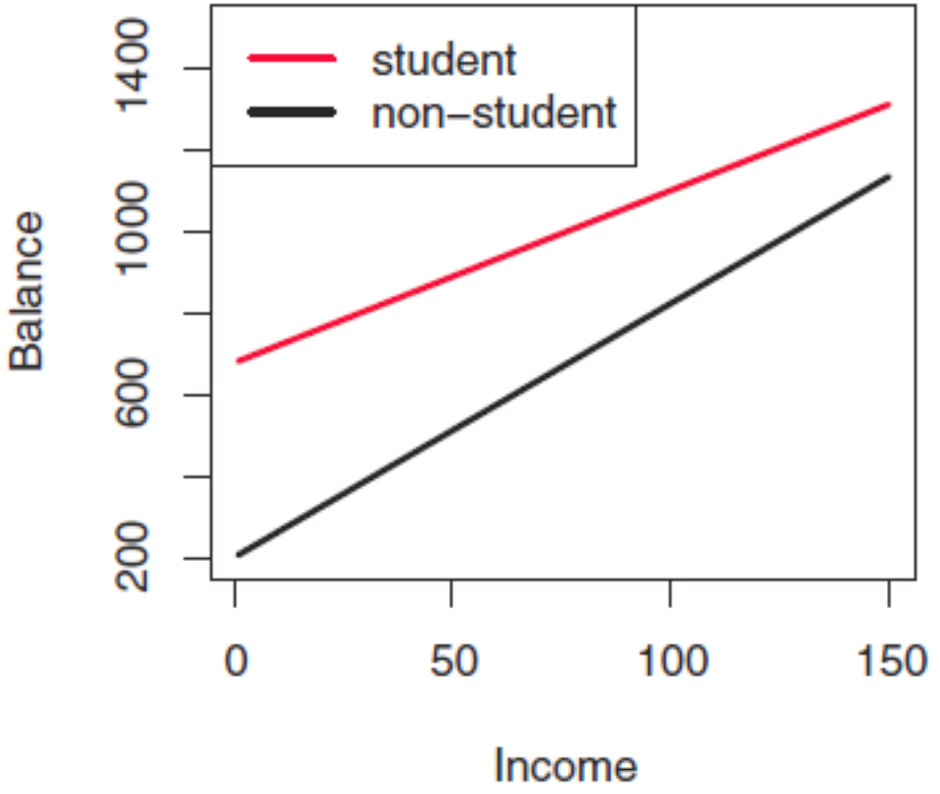
$$y_i = \alpha + \beta_1\{1\ if\ student\} + \beta_2 income + \epsilon_i$$



Understand binary features

β_1 allows for a different y-intercept for students

β_3 allows for a different Slope for students



$$y_i = \alpha + \beta_1\{1\ if\ student\} + \beta_2 income + \beta_3\{1\ if\ student\} * income + \epsilon_i$$

Interaction terms

Interaction term: multiplying two features by each other. When done as such:
{continuous feature}*{binary feature}

it allows for a different slope for the group represented by the binary feature. Note, be sure to include the continuous feature without the interaction as well.

Example, suppose temperature has a different impact on # of bikeshare trips taken on weekdays vs. weekends. We can create a binary variable `is_weekend`.

If we add `is_weekend` to the model, we allow for a different baseline ridership on weekends. If we add `is_weekend*temp`, we allow temperature to have a different effect for weekends.

We can "interact" `is_weekend` with a polynomial in `temp`. This effectively gives us a new model for temperature for the weekends vs. weekdays

Feature explosion

If we have two raw features, A and B, how many models can we make?

Without interactions (4): none, just A, just B, A + B
With interactions (8): A*B, A+B + A*B, A + A*B, B+ A*B

If we have ***p*** features then we have 2^{p+1} possible models!

$$p = 29 \longrightarrow 1{,}073{,}741{,}824$$

We cannot possible run all these models… so we'll learn methods to guide us.

Human intelligence is often a great guide as well.

Interpreting regression output

`Summary(my_model)` gives the coefficient estimates (beta's), t-statistics, standard errors and p-values.

t-statistics evaluate the hypothesis that the coefficient is equal to zero. Thus t greater than 1.96 in absolute value allows us to reject this hypothesis at the 95% level. P-values simply convert t-stats into the probability we would get something this far from zero due to sampling chance alone.

Note that t-statistics evaluate features "one by one". Overall model fit is a better guide for explanatory power (e.g. we might be better for leaving insignificant features in sometimes)

t-stats can be a good guide to throw out irrelevant features

Parametric vs. non-parametric regression

$$Y = f(X) + \epsilon$$

Parametric: f defined by a model we write down with *parameters that we have to estimate* (e.g. $\alpha, \beta_1, etc.$)

Non-parametric: we directly fit the data

Local averaging

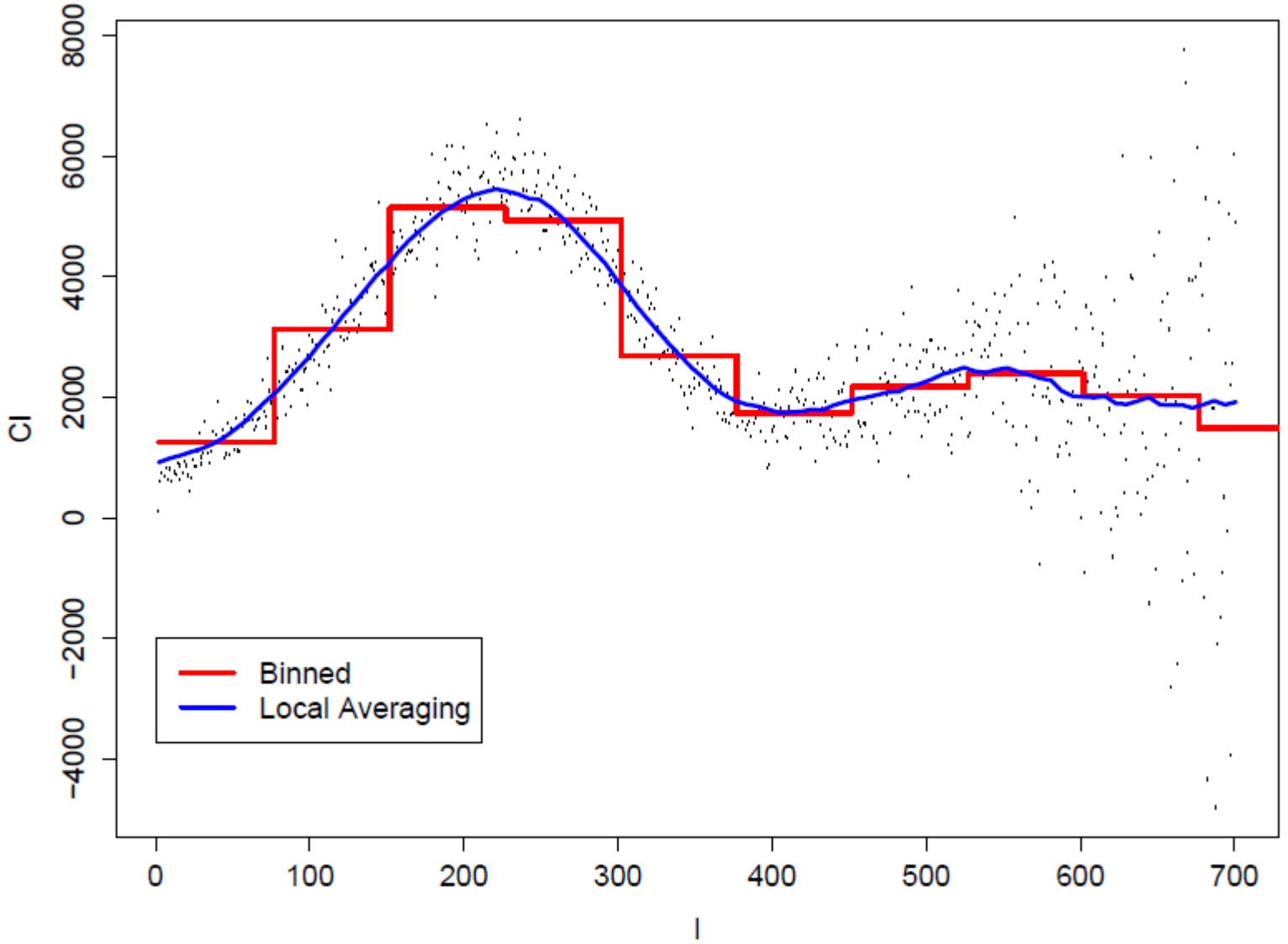
For each x let

$$N_x = \left[x - \frac{h}{2},\ x + \frac{h}{2}\right].$$

This is a moving window of length h, centered at x. Define

$$\widehat{f}(x) = \text{mean}\left\{Y_i :\ X_i \in N_x\right\}.$$

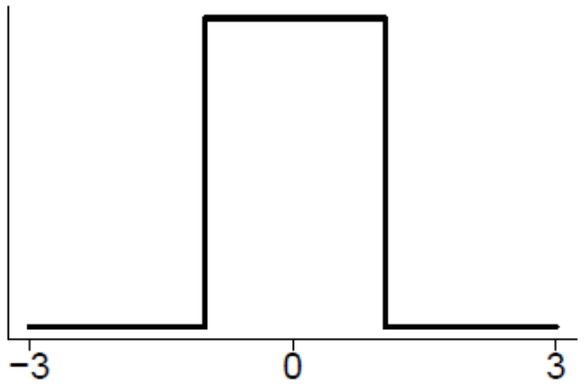
Local averaging vs. bins

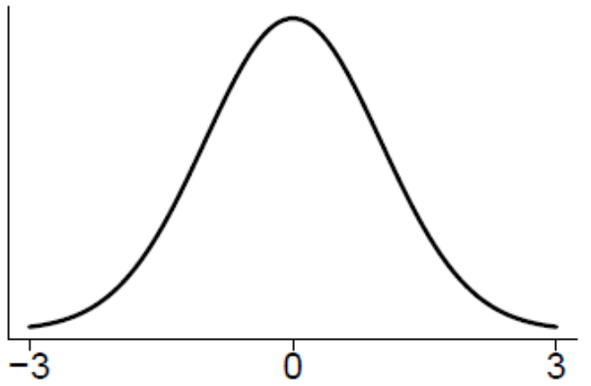


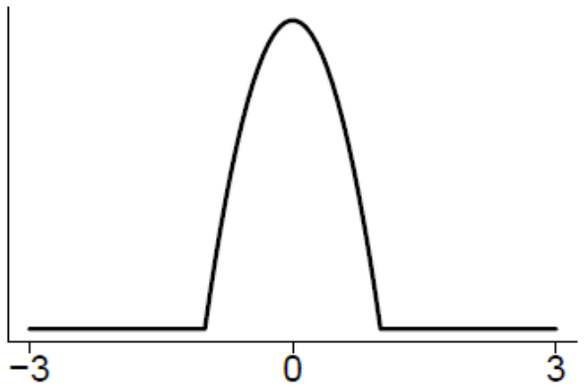
Kernels

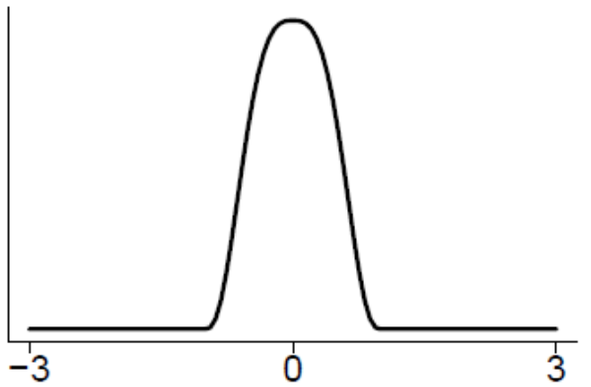
A method of assigning higher weight to the points closer to the target point I am trying to fit.

Choice of kernel usually does not matter much.









Bandwidths

Bandwidths tell us how wide to make our kernel (window)

Larger bandwidths → smoother functions because there is more data used to make the averages

Bandwidth is sometimes called the smoothing parameter.

Bandwidths matter!

K-nearest neighbors

An alternative to bandwidths is instead of specifying a fixed window, we can say “use the nearest K nearest neighbors”

This ensures each bin has the same amount of data and can be a useful tool

Functionally it will often be quite similar to using a bandwidth, except when there are “sparse data” issues (then K-NNN is preferred)

Often expressed as a fraction of data. E.g. NN=.1 says “each bin is 10% of data”

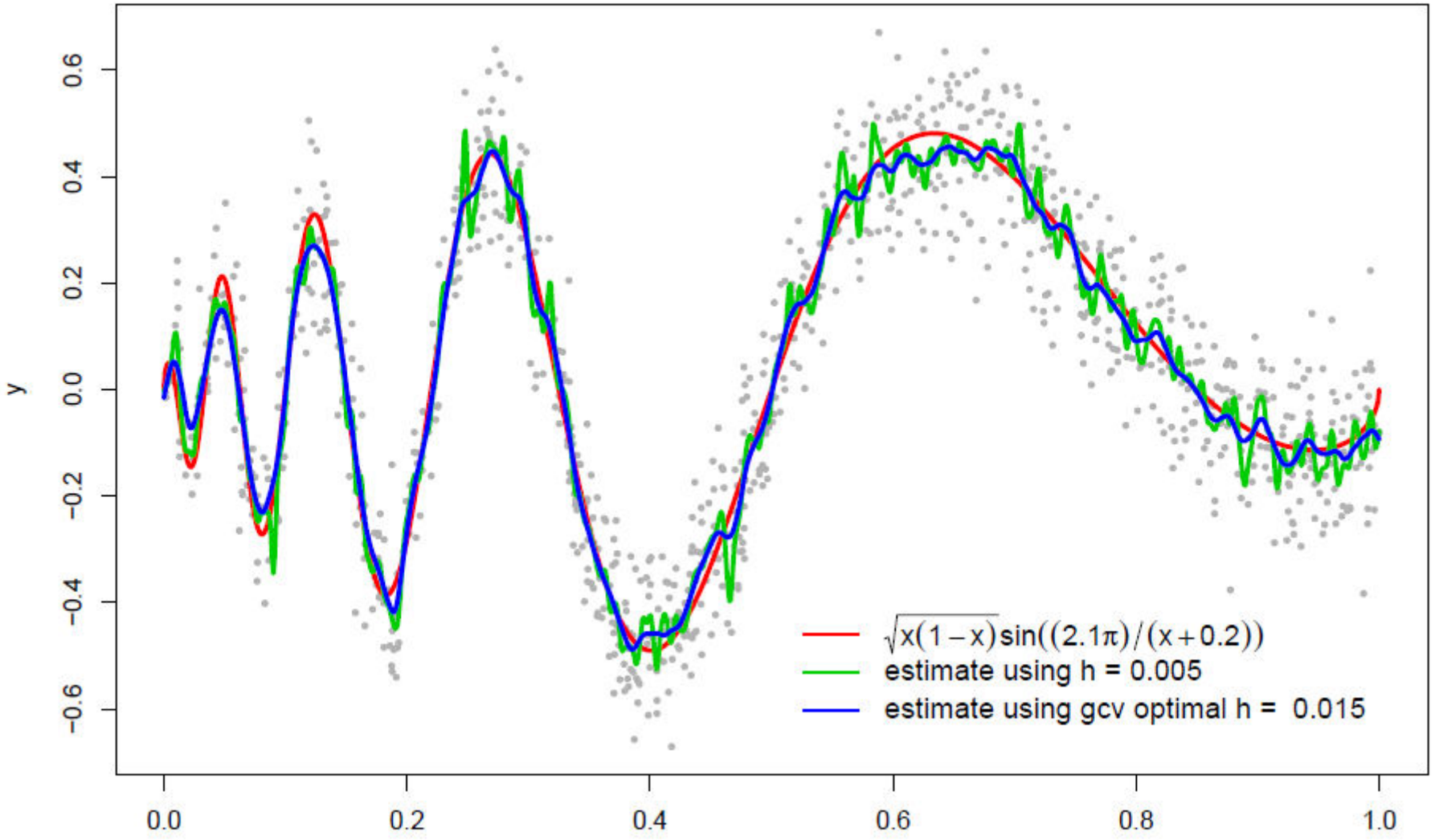
Local polynomial regression

Thing of local averaging as fitting a constant for a given window of data

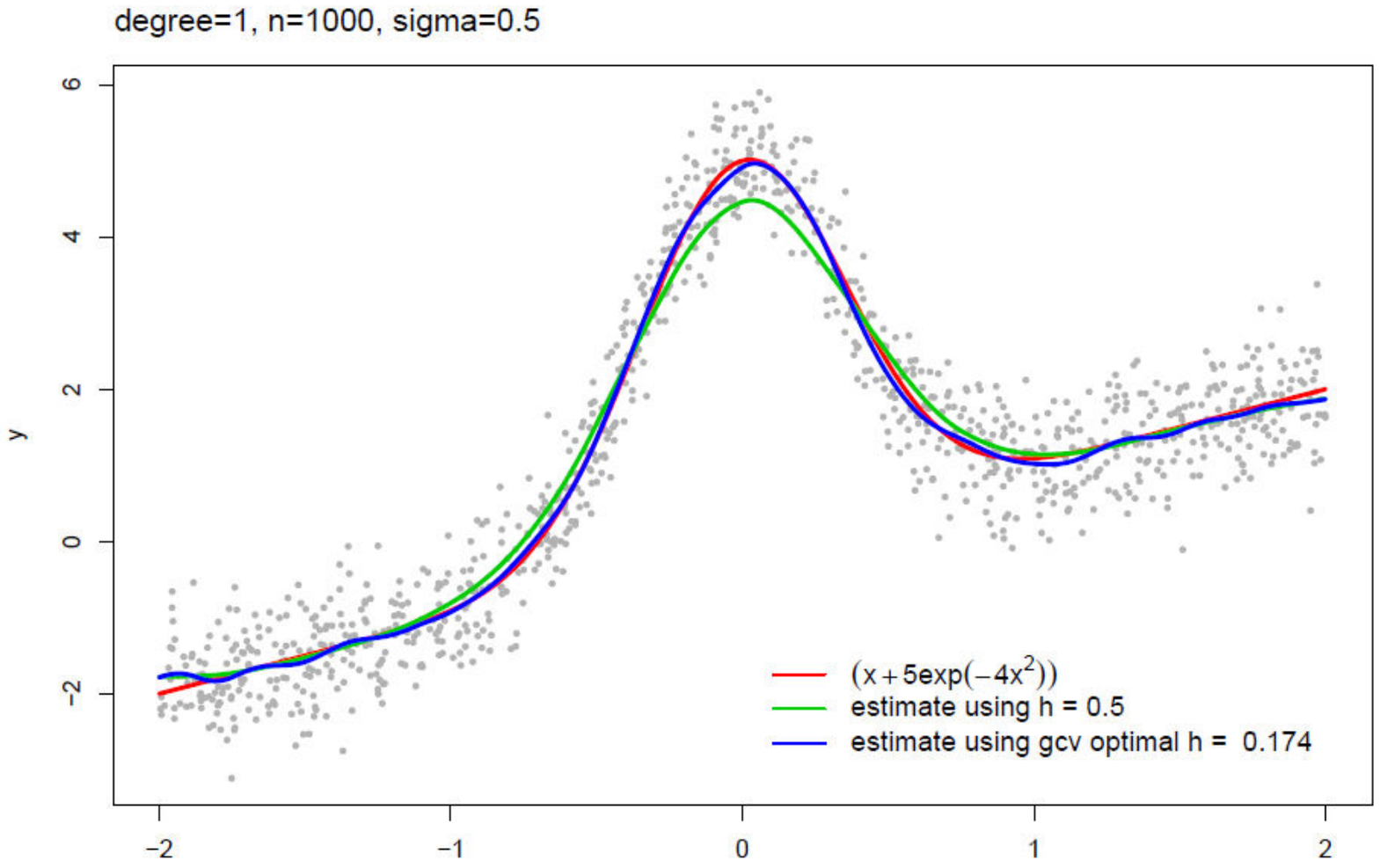
Instead of fitting a constant, we can fit a line (y=a*x + b) or a polynomial (y=a*x + b*x^2 + c), etc.

It is thus more general than local averaging, but very similar

Some examples



Some examples



Local polynomial in R `locfit`

```
install.packages(locfit)
library(locfit)
```

Modeler needs to specify: bandwidth (or fraction of nearest neighbors), degree of the local polynomial

Contrast: in parametric regression, we had to write down a model

Aside: we'll learn that non-parametric methods struggle in higher dimensions

When to use each model?

If there are a relatively small number of features (e.g. <4) then non-parametric makes a lot of sense.

With many features, the “curse of dimensionality” sets in and non-parametric methods fall apart (the “neighborhood” is generally empty)

Parametric models using interaction terms and polynomials are the preferred method with many features

“Semi-parametric” methods combine elements of both