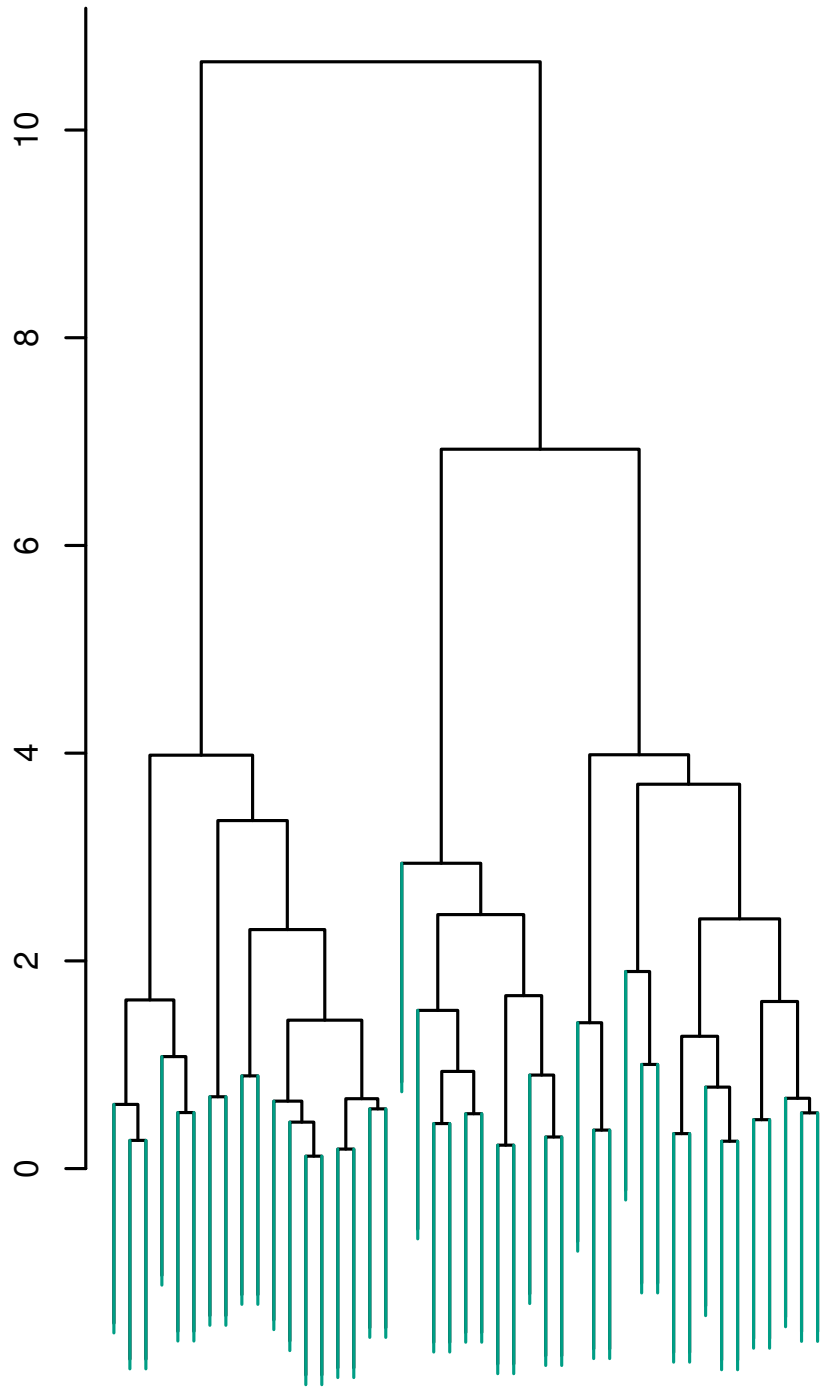
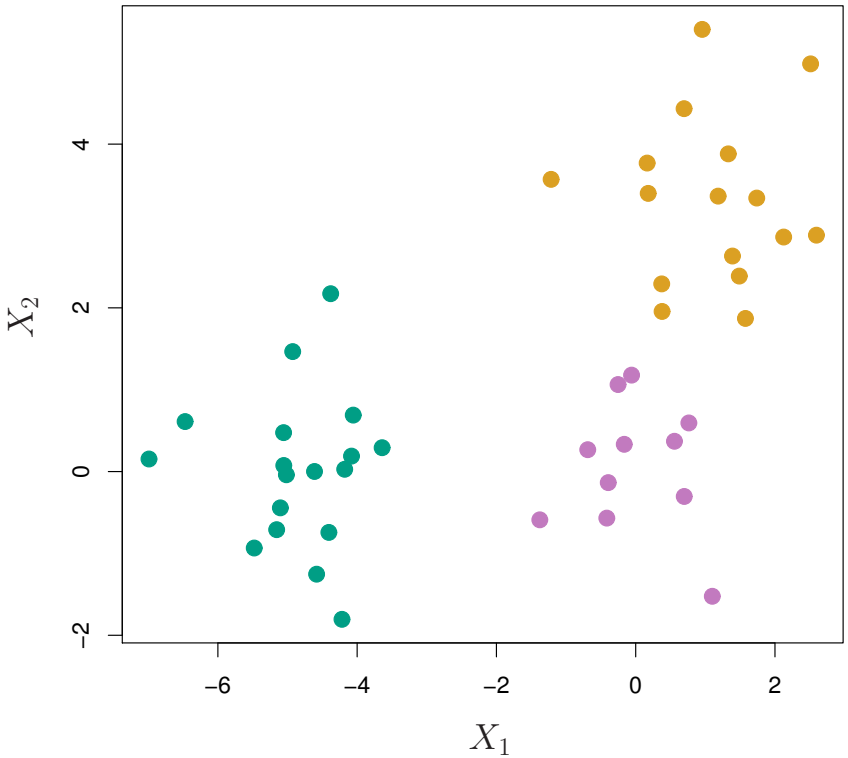
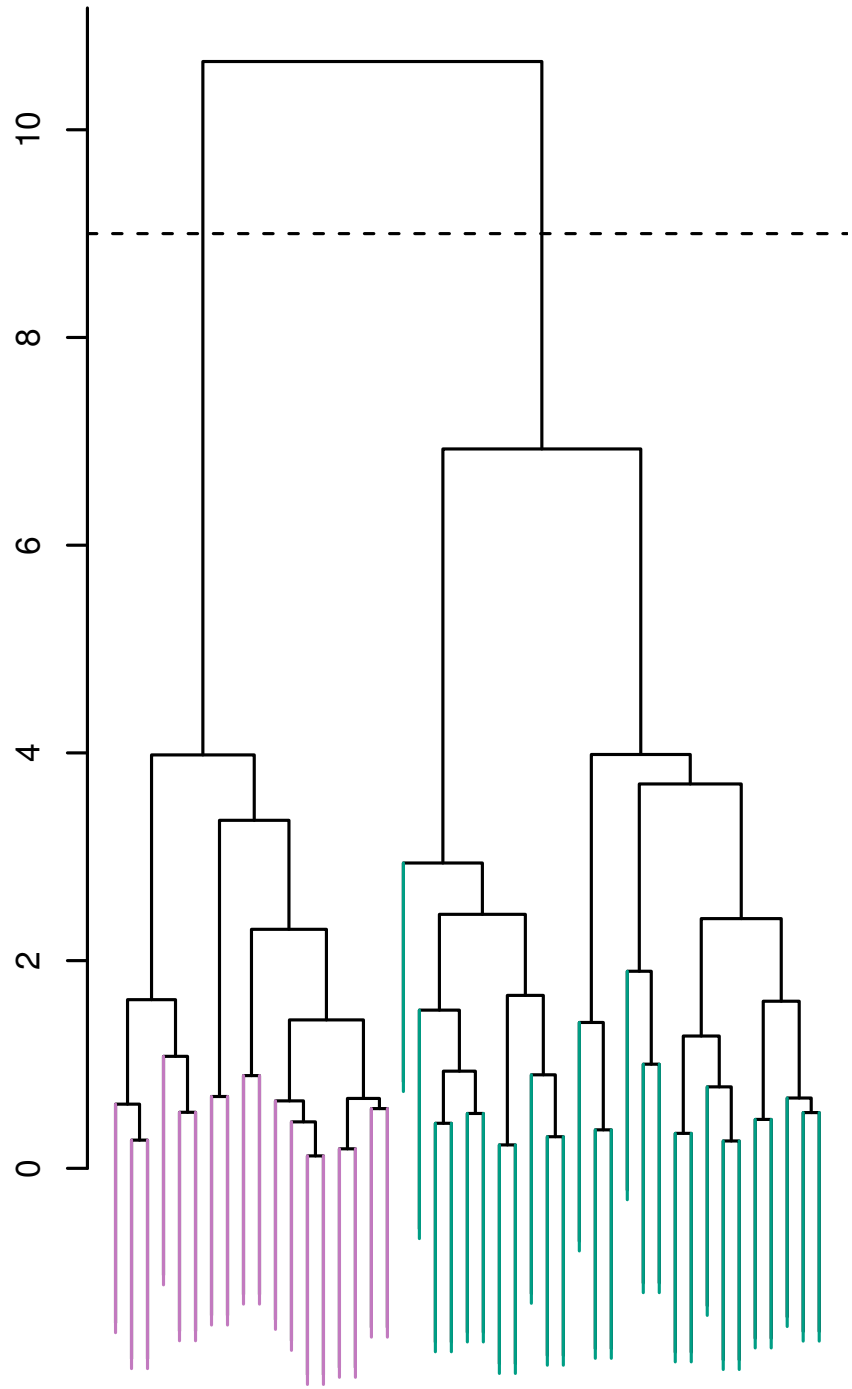
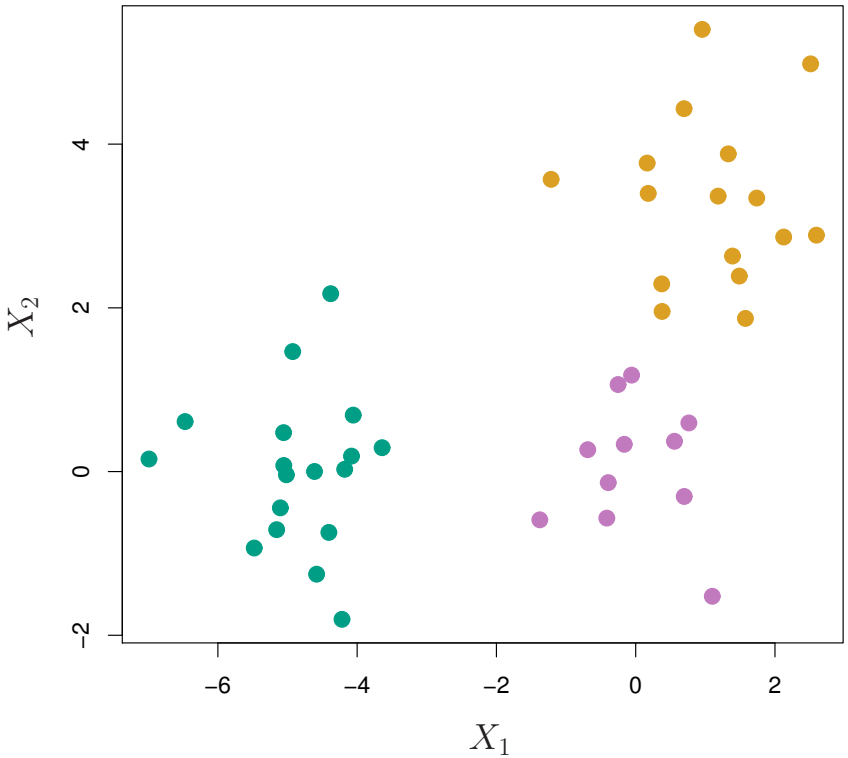
Unsupervised Learning II: Hierarchical Clustering

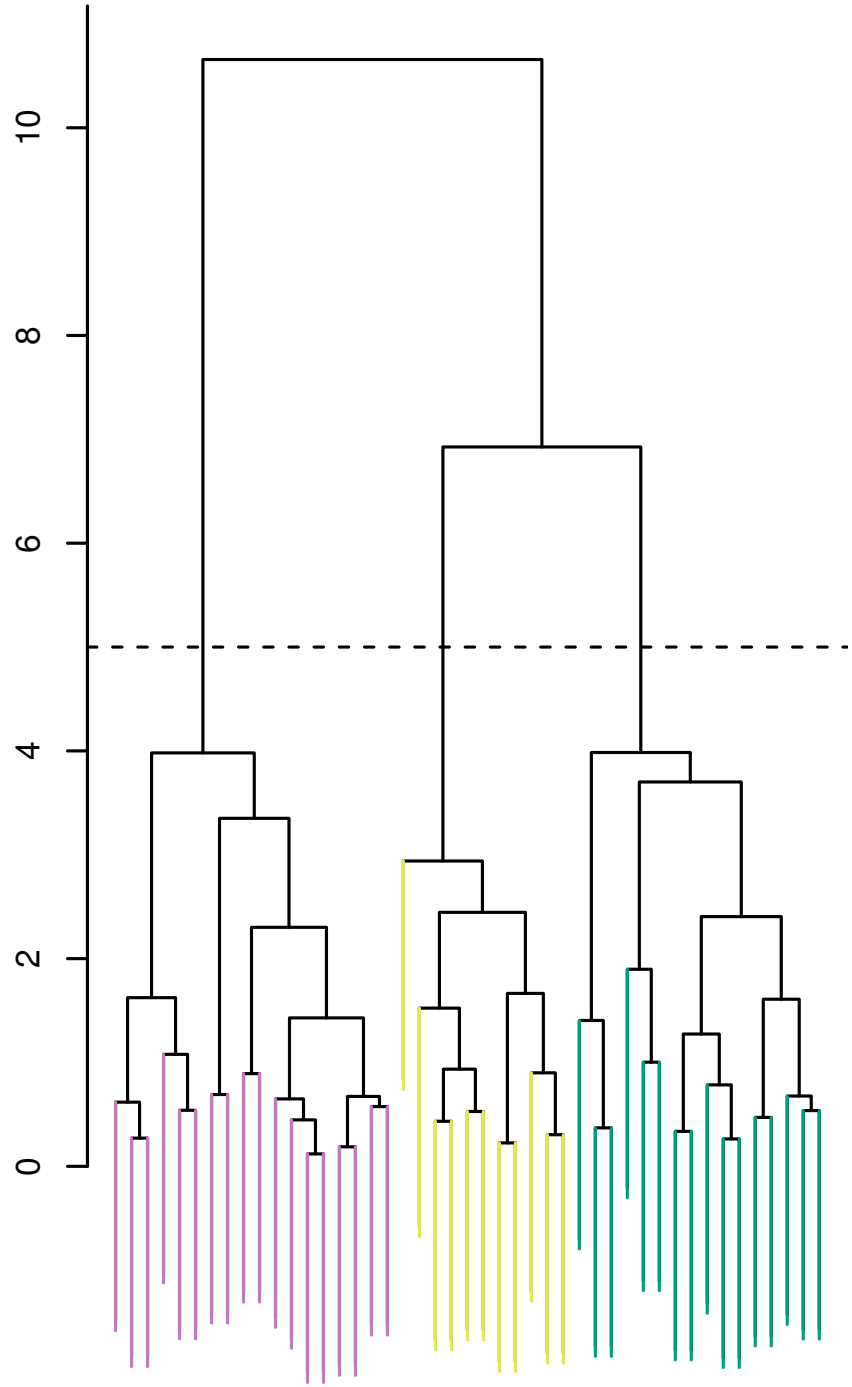
Yudong Chen
School of ORIE, Cornell University
ORIE 4740 Lec 15


0
2
4
6
8
10


X_2
4
2
0
−2
−6
−4
−2
0
2
X_1


10
8
6
4
2
0


X_2
4
2
0
−2
−6
−4
−2
0
2
X_1


10
8
6
4
2
0

The Hierarchical Clustering Algorithm

Input: n observations and a dissimilarity measure (e.g. Euclidean distance)

1. Compute all $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities.
   Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

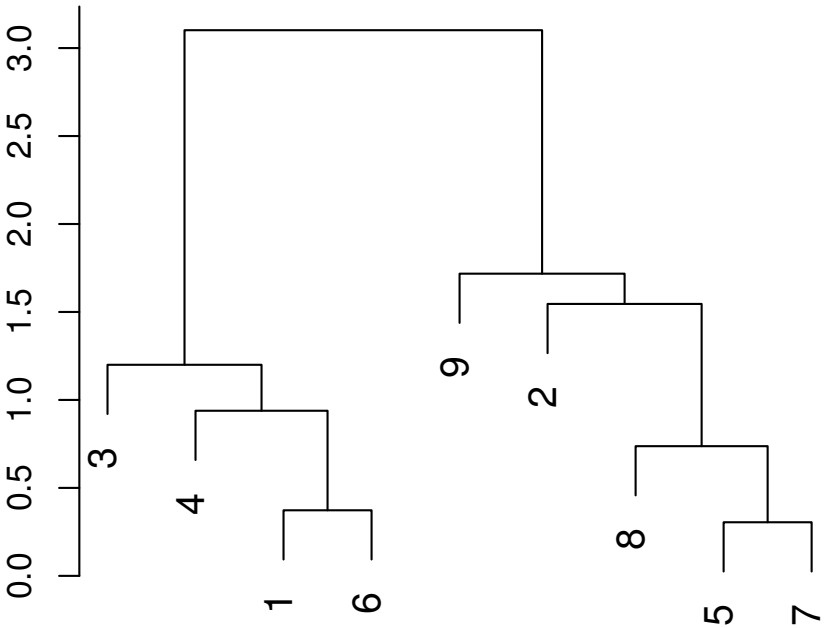
   a Among all pairs of the i clusters, identify the pair of clusters that are least dissimilar. Fuse these two clusters.

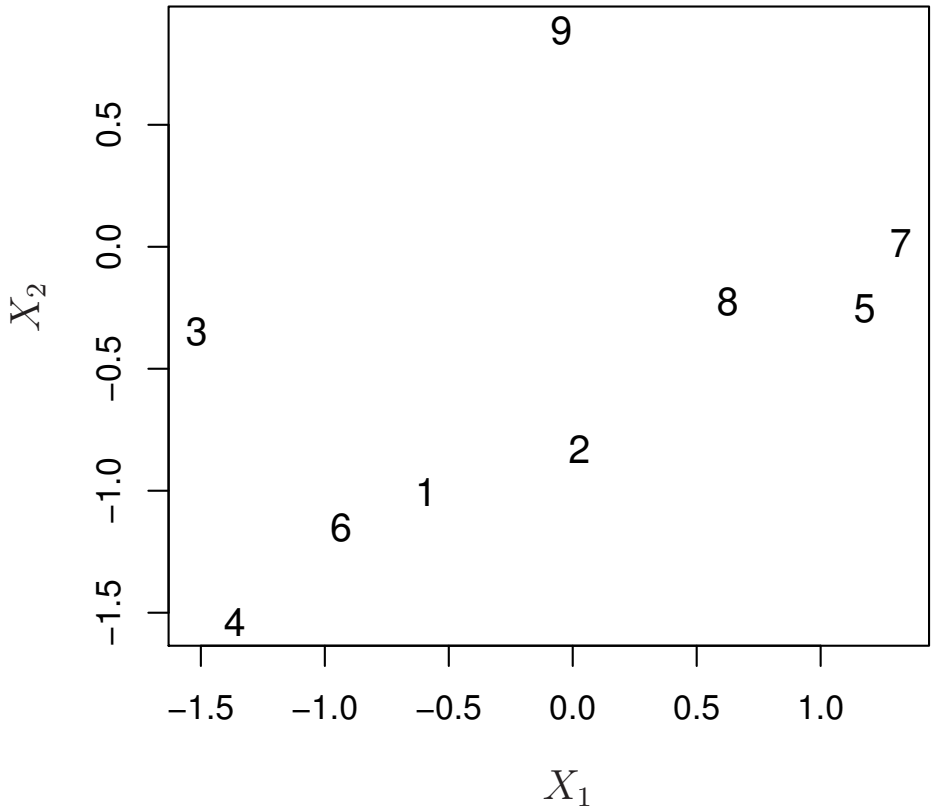
   - The vertical axis of a dendrogram: the distance/dissimilarity of the two clusters beimg fused.

   b Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

- Need a distance/dissimilarity measures between two clusters

Dendrogram





- Vertical axis: The distance/dissimilarity of the two clusters being fused.
- A large vertical jump: A clear sub-cluster
- Closeness in the horizontal axis: means nothing

Distance/Dissimilarity measures between clusters

Need a distance measure $D(C, C')$ between two clusters C and C'

- Based on a distance measure $d(x, y)$ between two observations x and y (e.g. Euclidean distance)

- Single linkage:

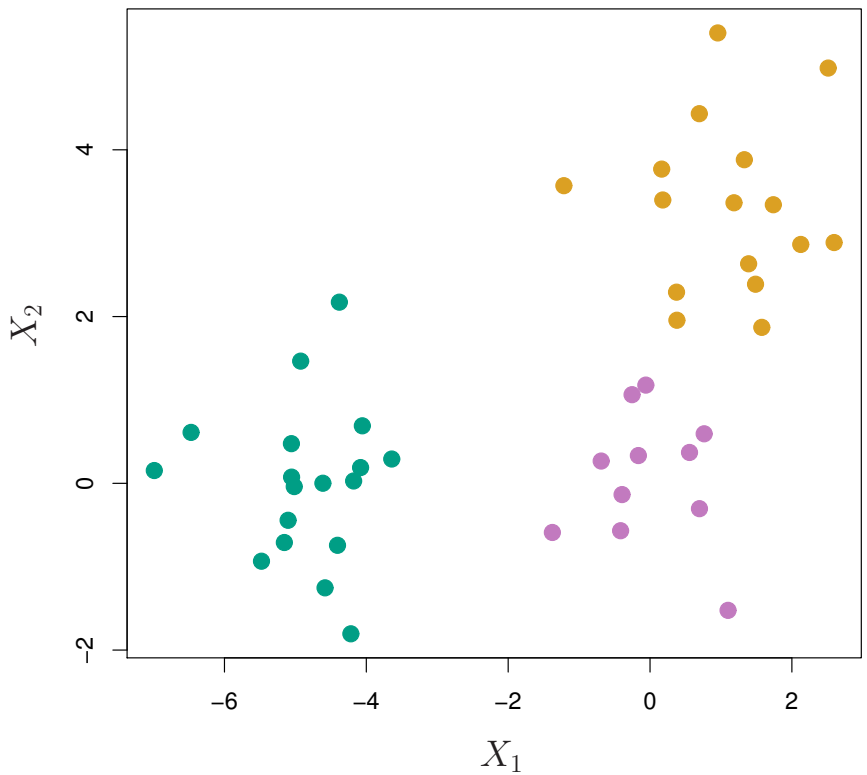
$$D(C, C') = \min_{x \in C, y \in C'} d(x, y)$$

- Complete linkage:

$$D(C, C') = \max_{x \in C, y \in C'} d(x, y)$$

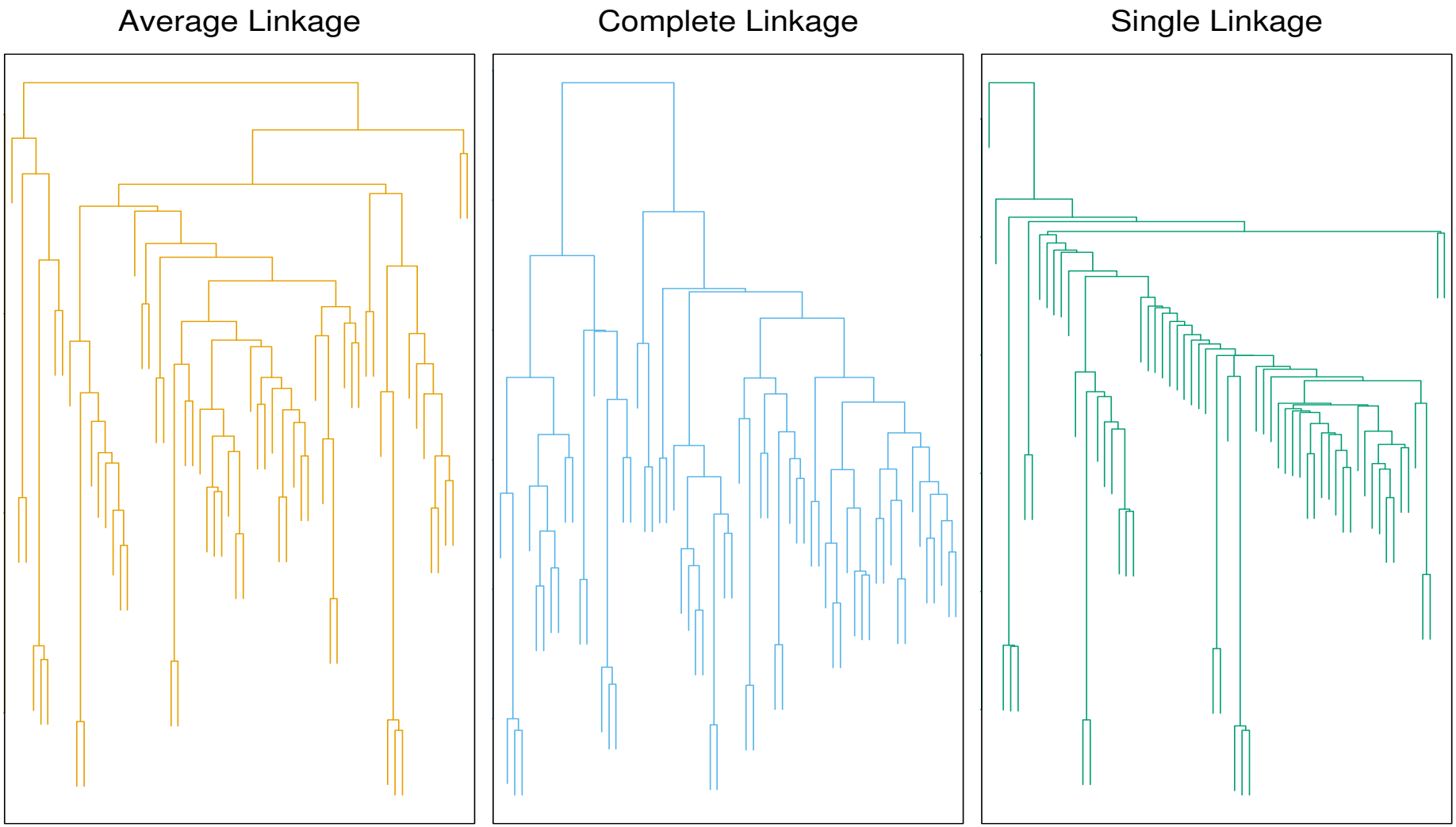
- Average linkage:

$$D(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{x \in C, y \in C'} d(x, y)$$



Choice of distance: Has a strong effect on the clusters

- Single linkage: Tend to produce unbalanced clusters
- Complete linkage: Popular; balanced, compact clusters
- Average linkage: Popular; in-between



Distance measures between observations (Optional)

Dissimilarity/distance b/w two observations x and y (p-dimensional vectors)

$$d(x, y)$$

- The building block for measuring distance b/w clusters
- Often more important than the choice of clustering algorithm

For categorical variables:

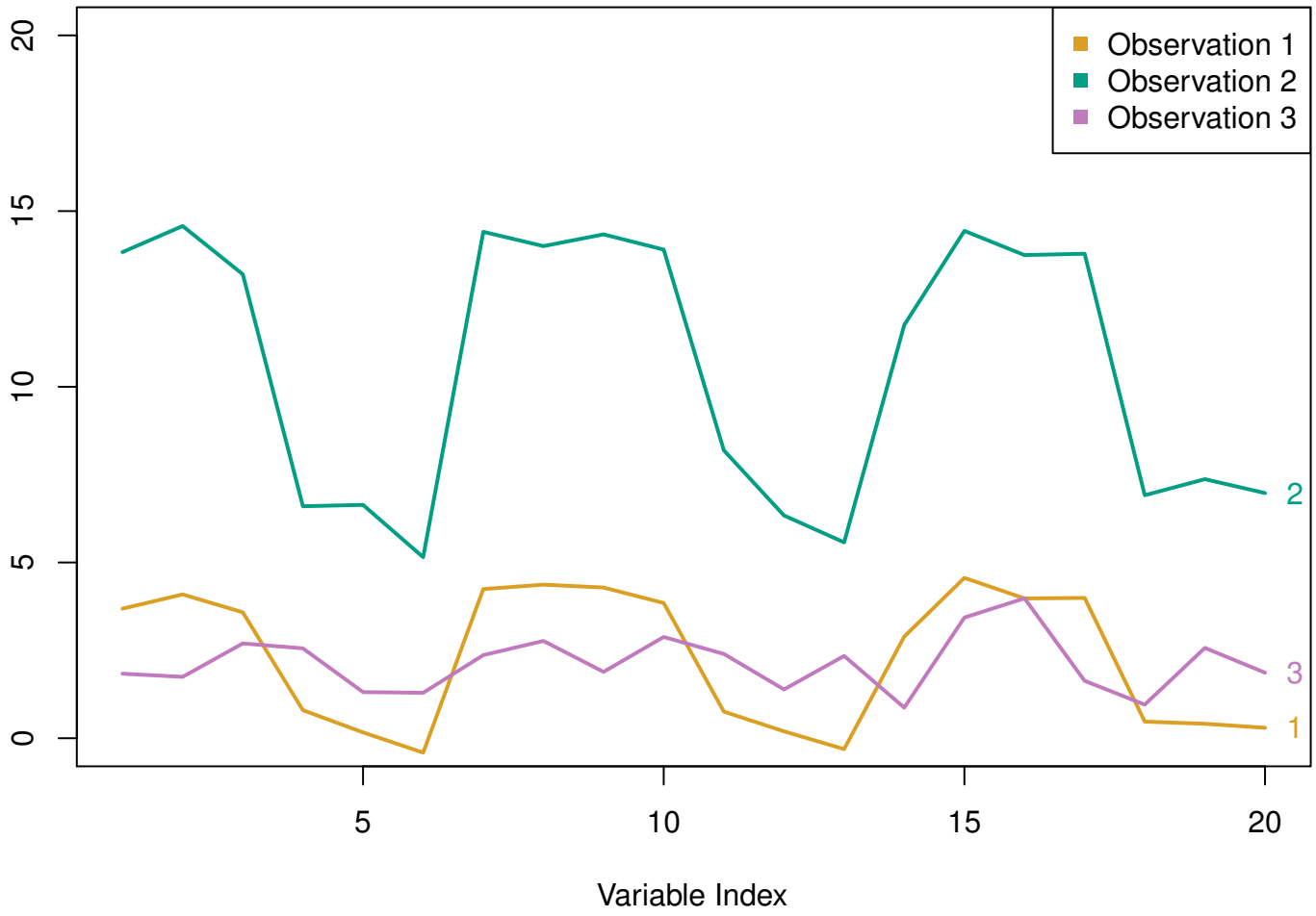
$$\sum_{j=1}^{p} 1(x_j \neq x'_j)$$

For continuous variables:

- Squared Euclidean (ℓ_2) distance: $\sum_{j=1}^{p}(x_j - y_j)^2$
- ℓ_1 distance: $\sum_{j=1}^{p} |x_j - y_j|$
- $\ell_{1/2}$ distance: $\sum_{j=1}^{p} |x_j - y_j|^{1/2}$
- Correlation-based distance: $1 - \text{corr}(x, y) = 1 - \cos \angle(x - \bar{x}, y - \bar{y})$

Distance measures between observations (Optional)

Euclidean distance vs. Correlation-based distance



Summary of Hierarchical Clustering

- Sequentially merge similar observations/clusters into larger clusters
- Produce a dendrogram: a hierarchy or clusters
- Cut the dendrogram to obtain clusters
- Need to specify distance measures
  - Distance b/w observations: Euclidean, Correlation-based
  - Distance b/w clusters: Average, Complete, Single
- Typically standardization is a good practice

An Engineer's Guide to Clustering

- Standardize variables (unless you have reasons not to)
- Try different options of
  - Algorithm: K-Means, Hierarchical
  - # of clusters K
  - Where to cut dendrogram
  - Observation distance: Euclidean, Correlation,...
  - Cluster distance: Average, Complete, Average
- To get a complete picture
- Find patterns that consistently emerge
- Visualizing your data
- Cluster results are not absolute truth
- Use domain knowledge!

Unsupervised Learning III: Principal Component Analysis

Yudong Chen
School of ORIE, Cornell University
ORIE 4740 Lec 16–17

Suppose that $X \in \mathbb{R}^{n \times p}$ is a data matrix with n observations and p predictors.

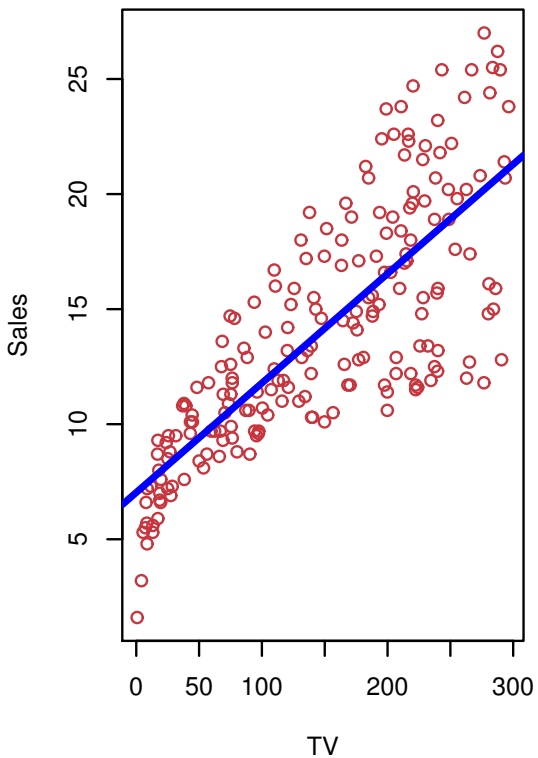
Which of the following is true?

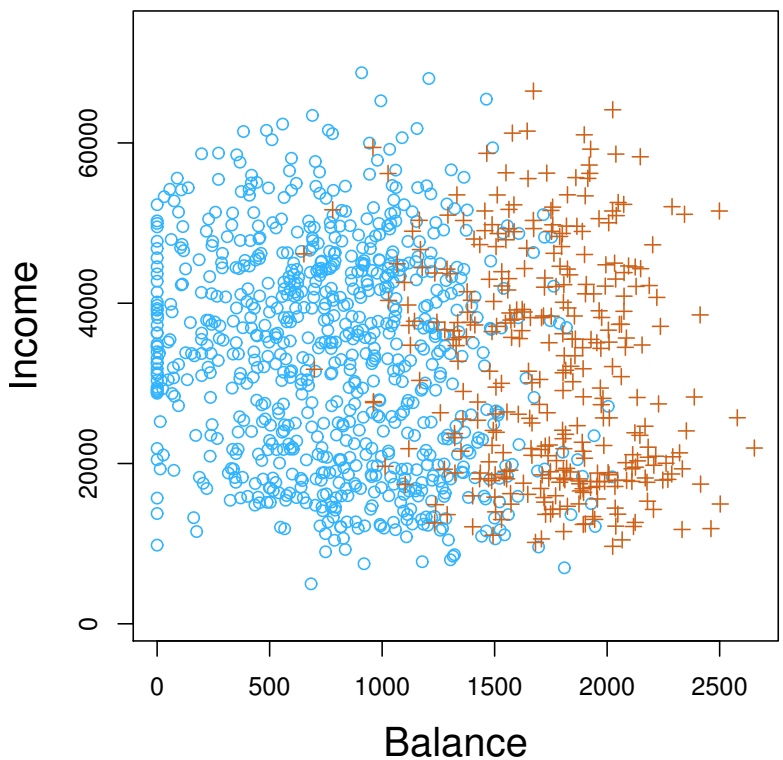
A. $X^\top X$ is a square matrix.
B. $X^\top X$ is a symmetric matrix.
C. All the eigen values of $X^\top X$ are non-negative.
D. If u^1 is an eigen vector of $X^\top X$ with eigen value λ_1, then $u^{1\top} X^\top X u^1 = \lambda_1$
E. All of the above.

Supervised Learning

- Learn a rule for predicting an response variable based on some predictor variables.
- Have a set of training data, in which the predictors and response values are known for each samples.

$$(x_{11}, x_{12}, x_{13}, y_1), (x_{21}, x_{22}, x_{23}, y_2), \ldots\ldots, (x_{n1}, x_{n2}, x_{n3}, y_n)$$





Unsupervised Learning

- A set of p variables/features measured on n observations

$$(x_{11}, x_{12}, x_{13}), (x_{21}, x_{22}, x_{23}), \ldots\ldots, (x_{n1}, x_{n2}, x_{n3})$$

- No associated response y

- Goal: Discover interesting patterns about the data

Unsupervised Learning

Often more challenging than supervised learning.

Difficulties:

- No simple goal (want to find “interesting patterns”)
- Contrast to supervised learning: predict the response
- No true answer (No Y)
- Difficult to assess model accuracy

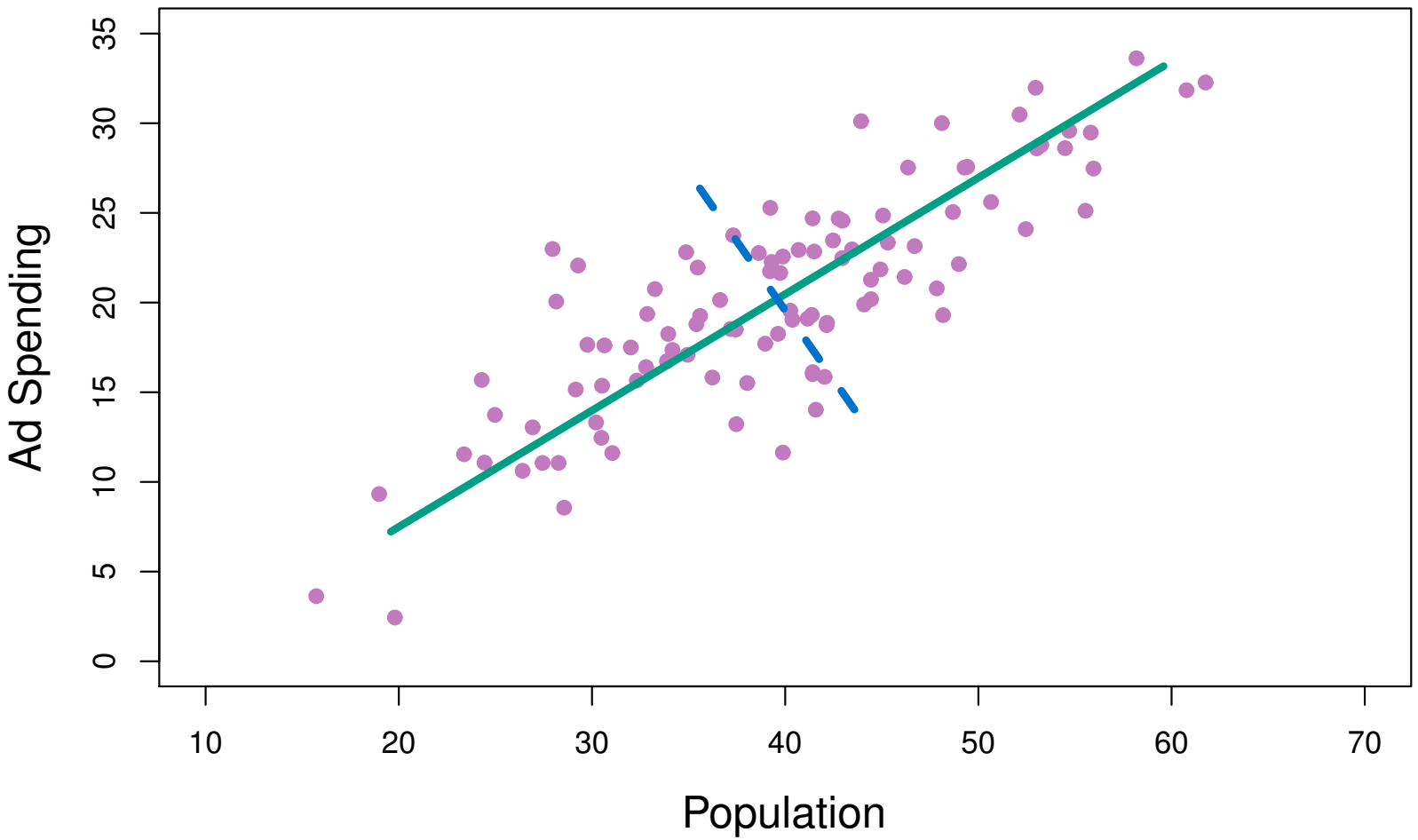
Used in exploratory data analysis

Principal Component Analysis

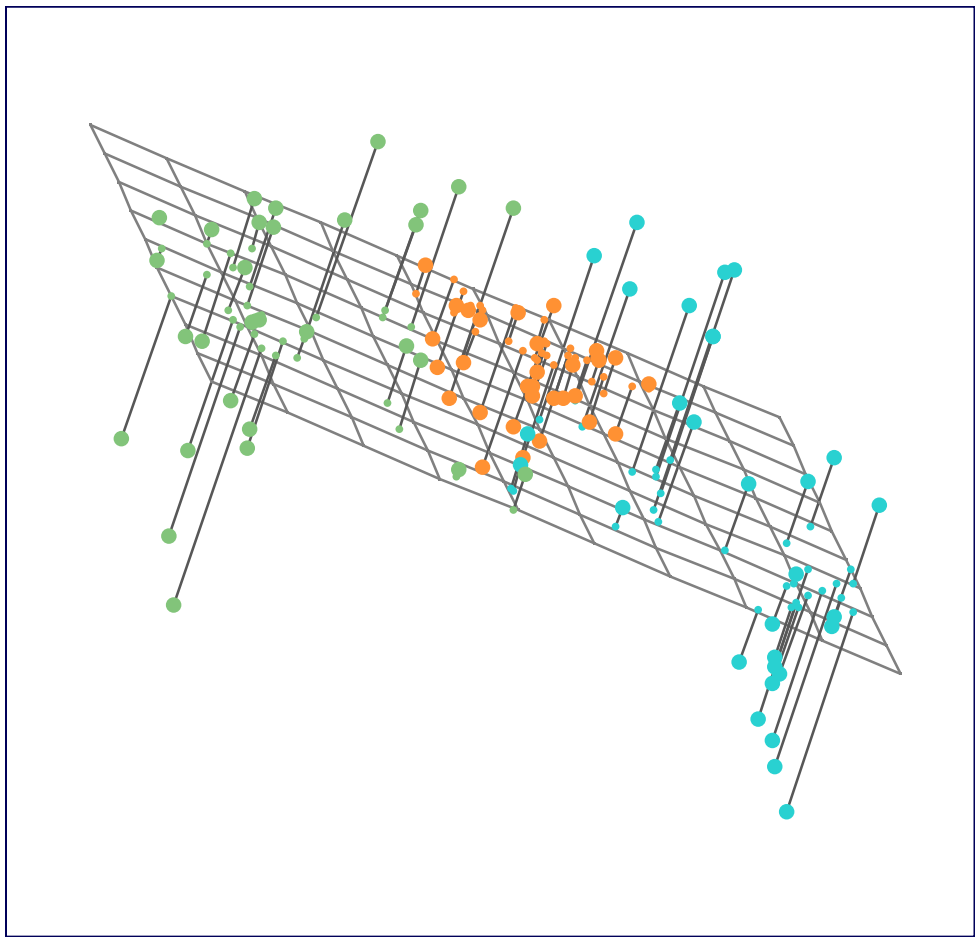
ISLR 10.2 (also cf. 6.3.1)

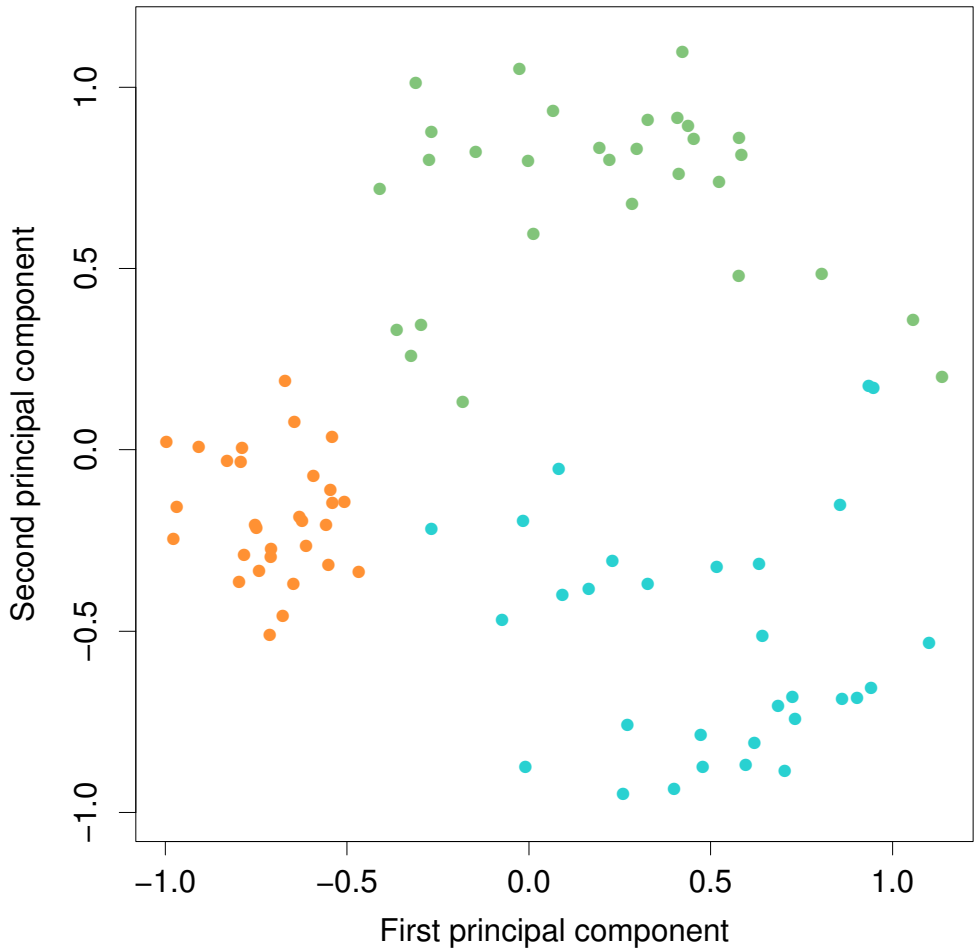
Find low-dimensional structures in the dataset

PCA in 2 dimensions



PCA in 3 dimensions





PCA

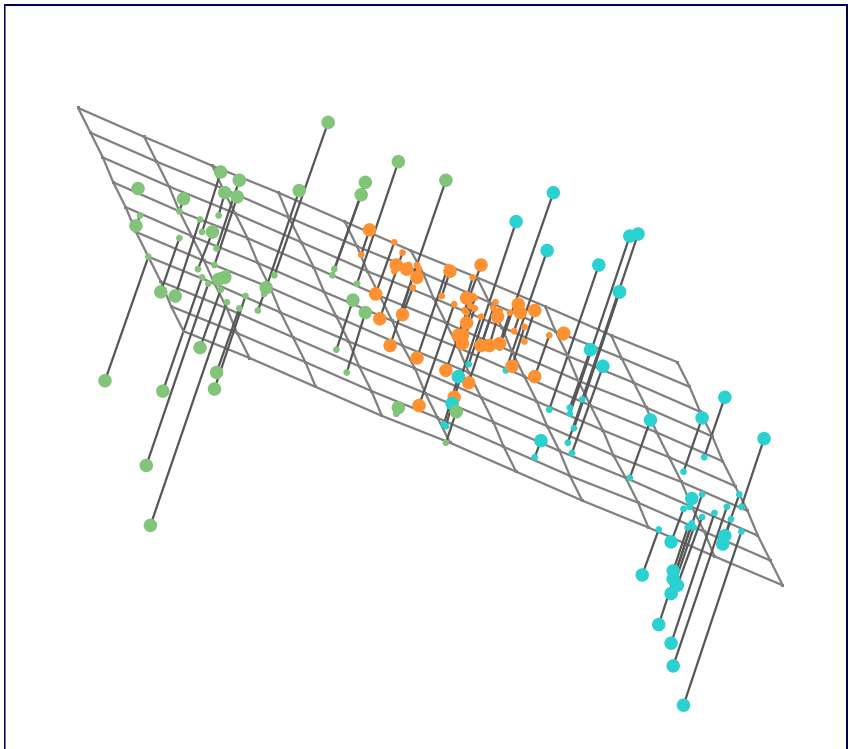
Idea: Find a succinct representation that best summarizes the data

Original data: p features/variables (p dimensional)

- Find a r-dim subspace on which the variance of the data is maximized
- (Equivalent) Find a r-dim subspace closest to the data

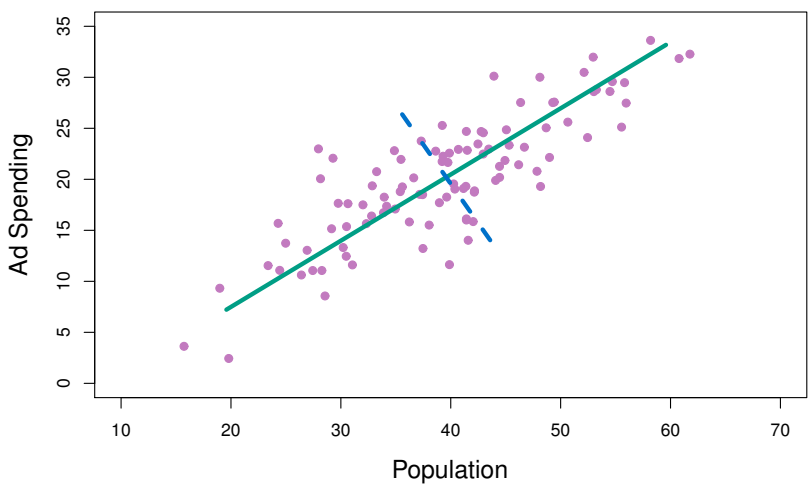
where $r < p$

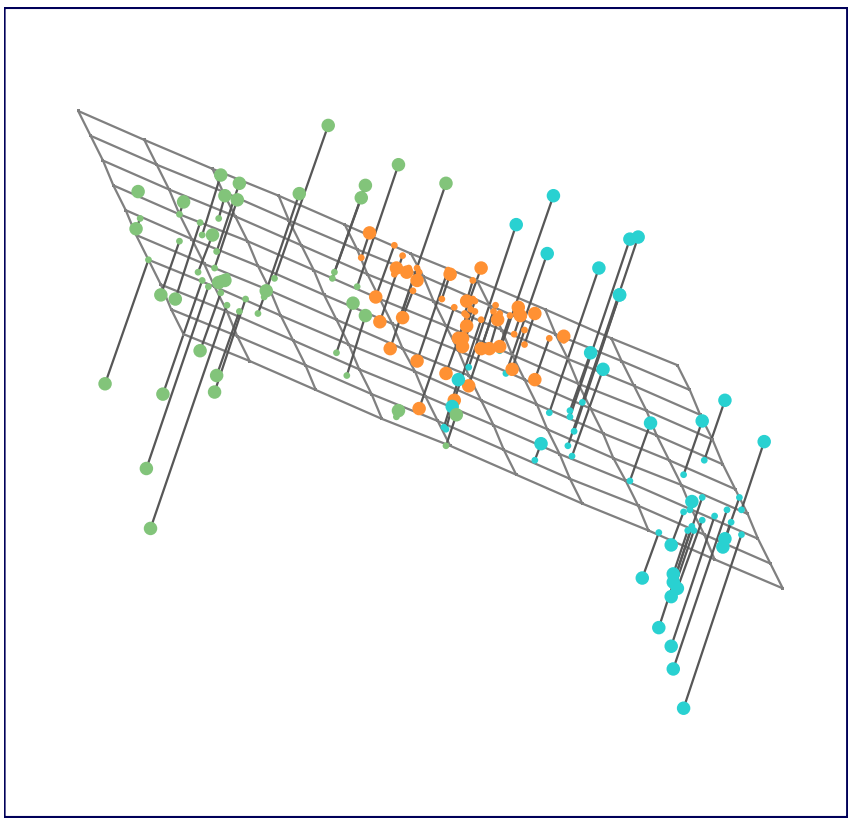
The first $r = 2$ principal components:



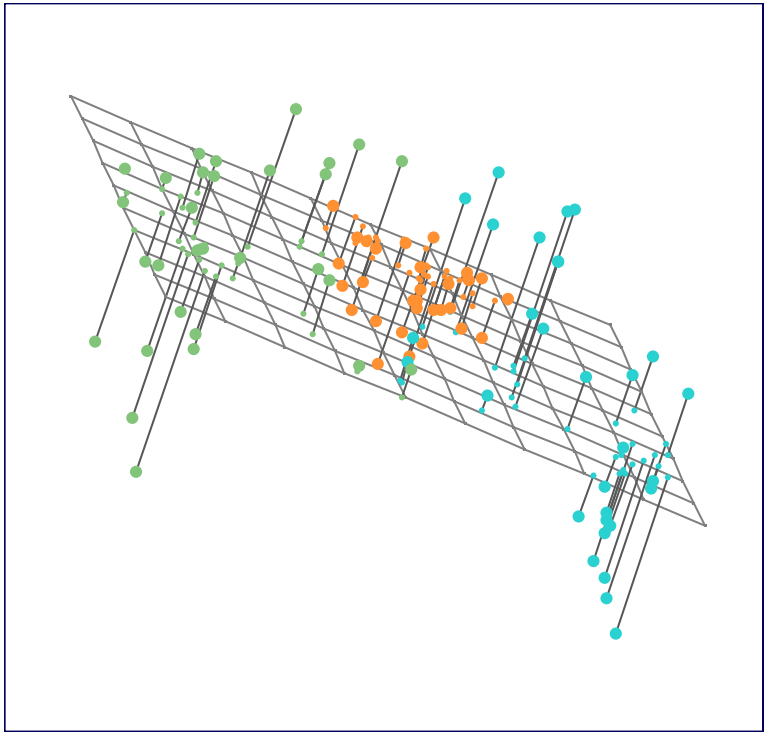
Principal Components

- $r = 1$: the first PC — a straight line
- $r = 2$: the first two PCs — a plane
- In general: the first r PCs — an r-dimensional subspace





Principal Components



A dimension reduction technique:

Reduce the original p-dim data to r-dim, such that (hopefully)

- Most of the information is kept
- Most of the noise is dropped
- Easier to store, manipulate and visualize

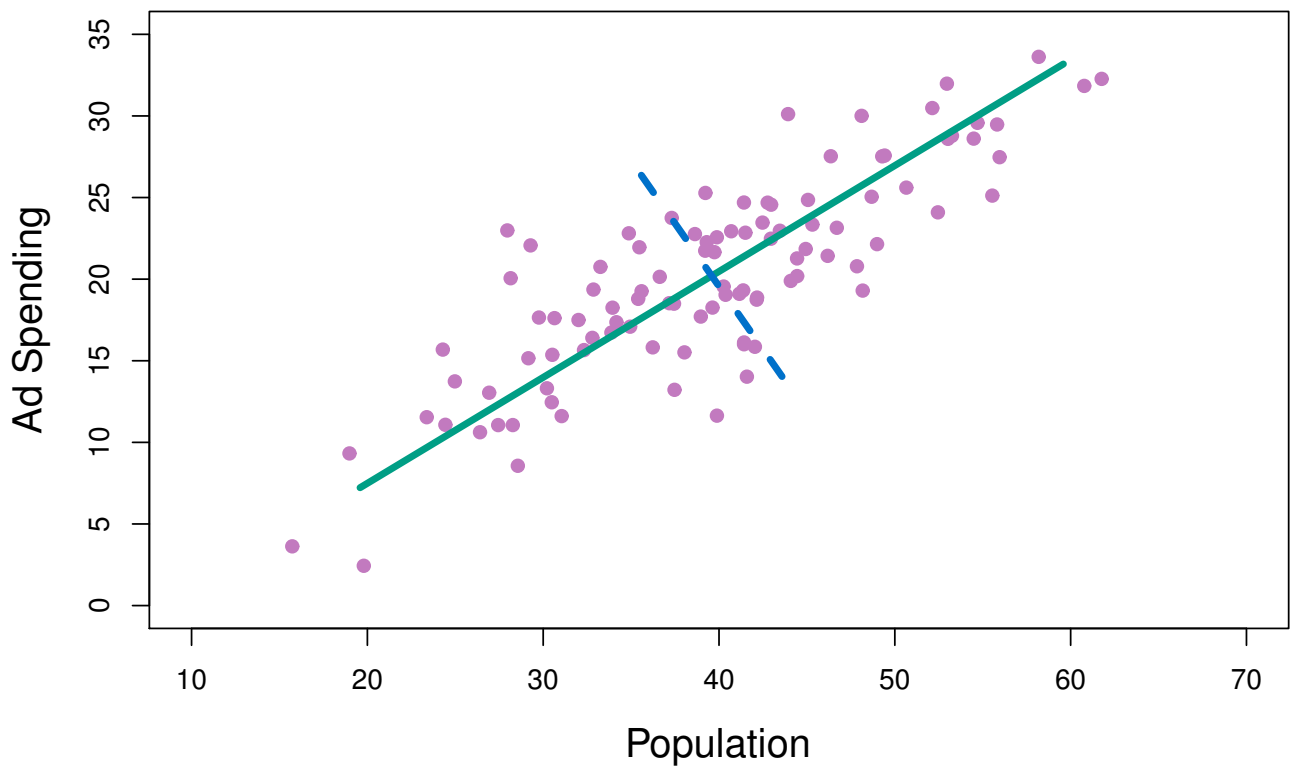
The First Principal Component: Mathematical Definitions

The 1st PC of features $X_1, \ldots, X_p$:

Linearly combine features in a way that retains the largest variance $\text{Var}(Z_1)$

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{p1} X_p,$$

where $\phi_{11}^2 + \phi_{21}^2 + \cdots + \phi_{p1}^2 = 1$



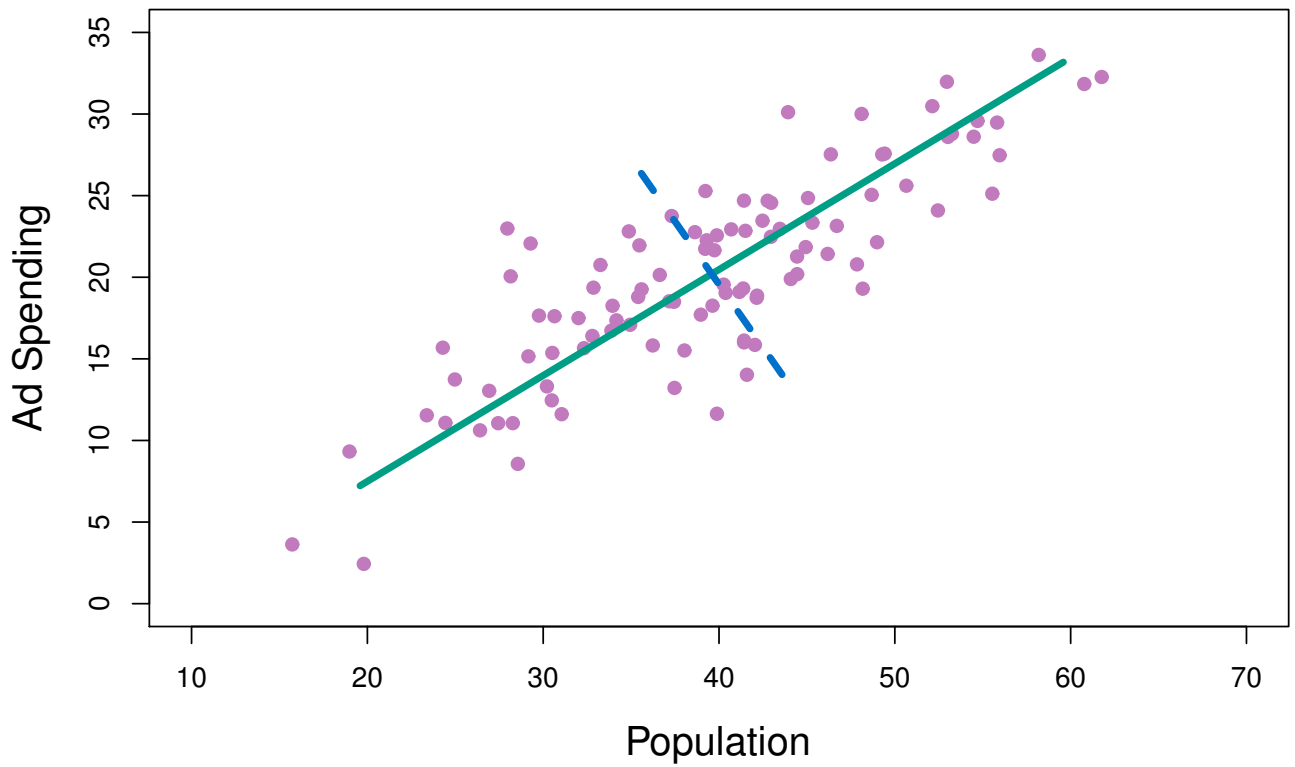
The First Principal Component: Loadings and Scores

Loadings: The direction of the line

- specified by p numbers $(\phi_{11}, \phi_{21}, \ldots, \phi_{p1})$

Scores: The projection of each observation onto this line

- specified by n numbers z_{i1}, $i = 1, 2, \ldots, n$



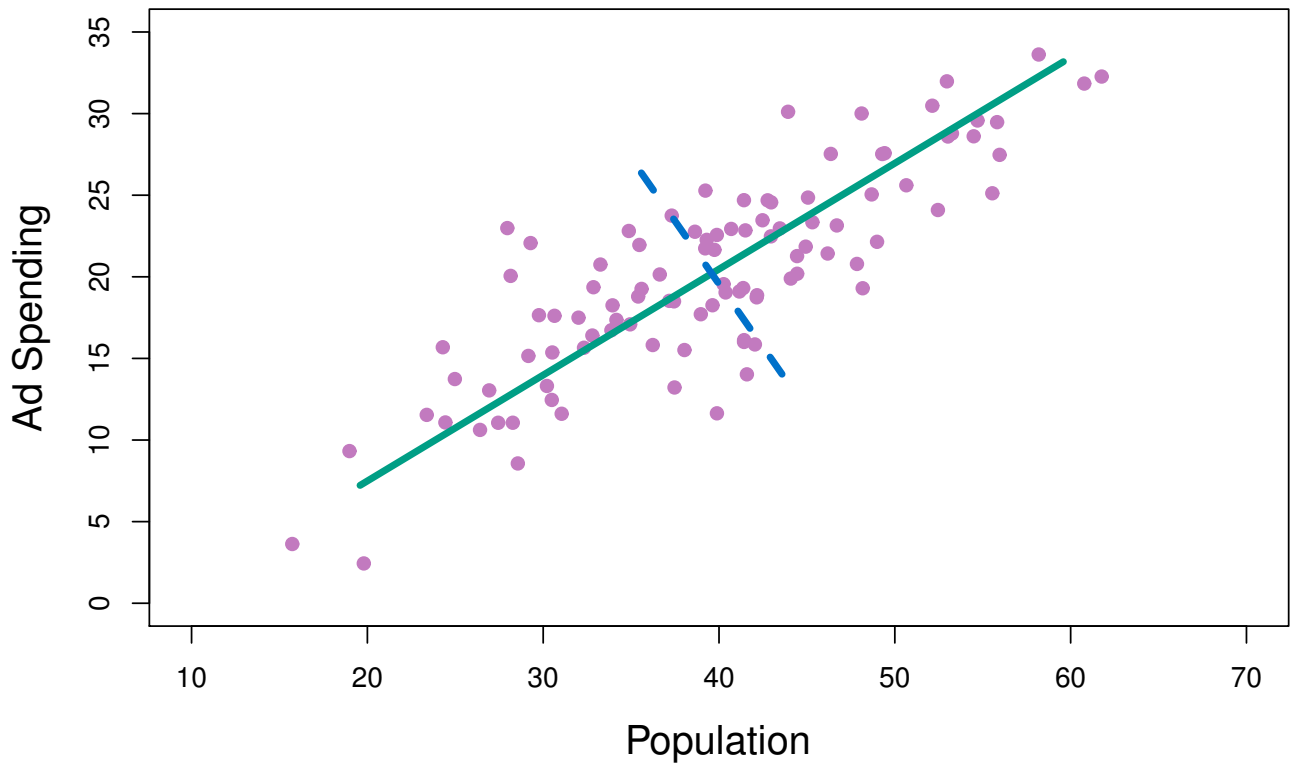
The Second Principal Component

The 2nd PC of features $X_1, \ldots, X_p$:

Linear combination of the features

$$Z_2 = \phi_{12} X_1 + \phi_{22} X_2 + \cdots + \phi_{p2} X_p,$$

with maximal variance, out of all combinations that are uncorrelated with Z_1

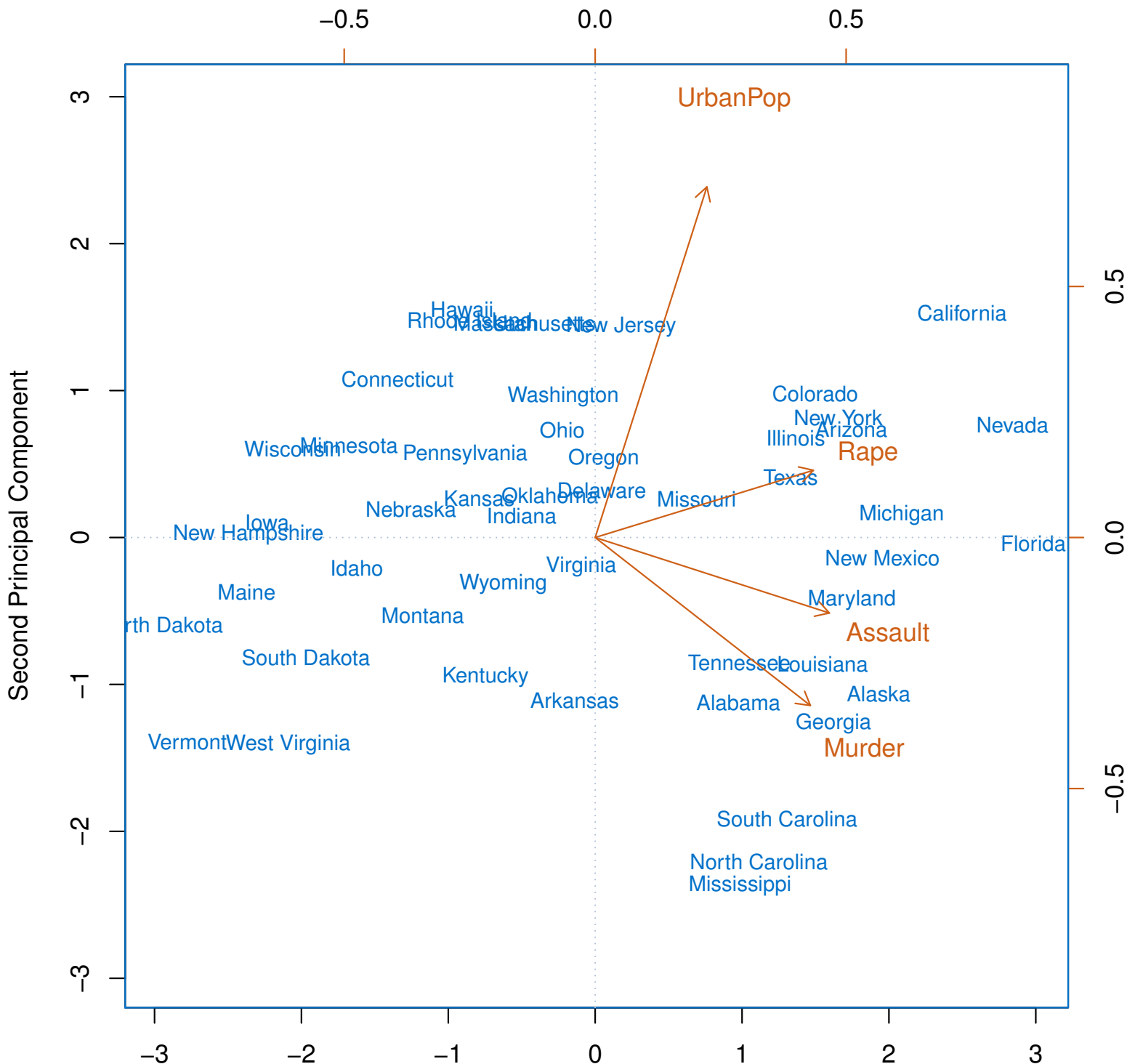


`USArrests` example

$p = 4$ features: `Assault, Murder, Rape, UrbanPop`

Find the loadings of the first 2 PCs:

	PC1	PC2
Murder	0.5358995	−0.4181809
Assault	0.5831836	−0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186


−0.5
0.0
0.5
UrbanPop
Second Principal Component
3
2
1
0
−1
−2
−3
0.5
0.0
−0.5
Hawaii
New Jersey
California
Connecticut
Washington
Colorado
New York
Ohio
Illinois
Arizona
Nevada
Wisconsin
Minnesota
Pennsylvania
Oregon
Rape
Texas
Delaware
Kansas
Oklahoma
Missouri
Nebraska
Indiana
Michigan
Iowa
New Hampshire
Florida
Virginia
New Mexico
Idaho
Wyoming
Maine
Maryland
Montana
Assault
South Dakota
Tennessee
Louisiana
Kentucky
Alaska
Arkansas
Alabama
Georgia
Murder
Vermont
West Virginia
South Carolina
North Carolina
Mississippi
−3
−2
−1
0
1
2
3

Choosing the number of principal components

In PCA, want to find directions that retains the most variance

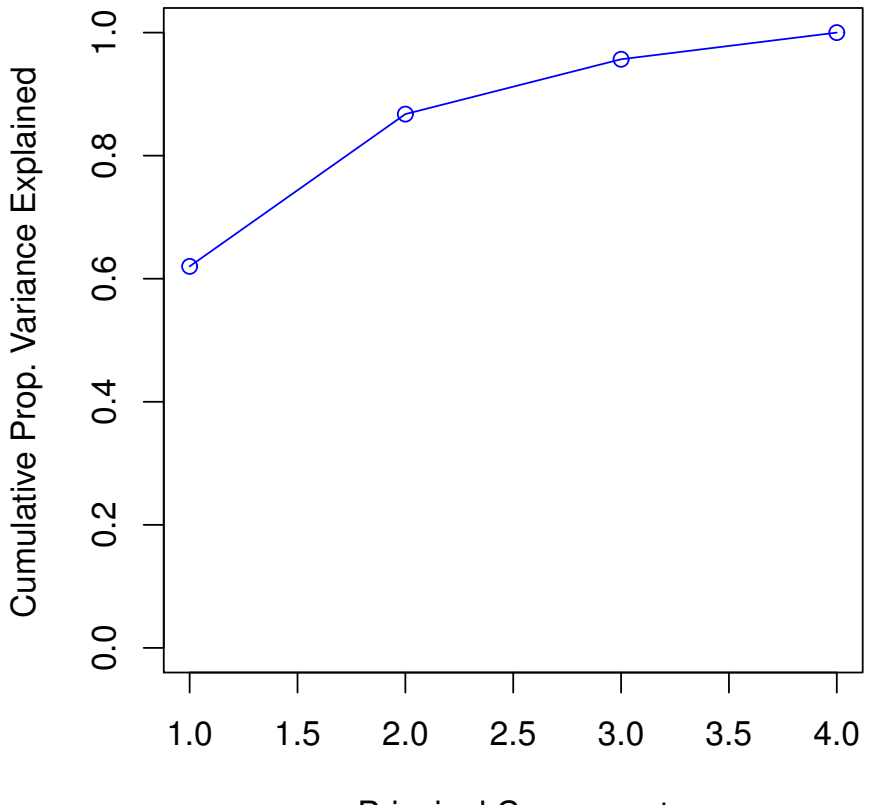
Proportion of Variance Explained (PVE) of the m-th PC direction

$$= \frac{\text{Var. explained by } m\text{-th PC}}{\text{total Var.}}$$

```
> pr.var = pr.out$sdev^2

> pve = pr.var / sum(pr.var)

> plot( cumsum(pve) )
```



PCA: Math and Computation (Optional)

Recall: For first PC, want to find the normalized loadings $\phi_{j1}, j = 1, 2, \ldots, p$ that maximizes the variance of the linear combination

$$z_{i1} = \sum_{j=1}^{p} \phi_{j1} x_{ij}$$

Mathematically: (assume x_{ij}'s centered)
Want to solve

$$\max_{\phi_{j1}, j=1,\ldots,p} \quad \text{Var}(\{z_{i1}\}) = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2$$

$$\text{subject to} \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

PCA: Math and Computation (Optional)

- Want to solve

$$\max_{\phi_{j1}, j=1,\ldots,p} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}\phi_{j1}^2 = 1.$$

- In matrix/vector notation:

$$\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\langle\vec{\phi}_1, \vec{x}_i\rangle^2$$

$$= (\vec{\phi}_1)^\top \underbrace{\frac{X^\top X}{n}}_{\text{Cov}(X)} \vec{\phi}_1 \qquad \text{(Lecture 25 Sec 5.1)}$$

- From notes on linear algebra:

This is maximized when $\vec{\phi}_1 =$ the first eigenvector of covariance matrix $\frac{X^\top X}{n}$

PCA: Math and Computation (Optional)

1st PC $\vec{\phi}_1$ = the first eigenvector of covariance matrix $\frac{1}{n}X^\top X$

Similarly.....

m-th PC $\vec{\phi}_k$ = the *m*-th eigenvector of covariance matrix $\frac{1}{n}X^\top X$

Nonlinear Methods

Yudong Chen
School of ORIE, Cornell University
ORIE 4740 Lec 18–19

Recap: What we covered so far

- Concepts: model flexibility; bias-variance tradeoffs
- Linear regression: fitting and evaluation models
- Classification: Logistic regression; KNN
- Model selection and regularization: subset selection; Ridge; Lasso; principal component regression
- Unsupervised techniques: PCA; k-means and hierarchical clustering
- Cross-validation

Recap: Supervised vs. Unsupervised

Supervised learning:

- Regression
- Classification
- Regularization & variable selection: apply to both
- CV: estimate test errors to choose models (tunning parameters)

Unsupervised learning:

- PCA
- Clustering

Recap: Linear vs. Nonlinear

Linear techniques:

- Linear regression
- Logistic regression
- k-means
- PCA

Simple extensions of linear techniques:

- Adding high-order and interaction terms
- Converting to dummy variables

Nonlinear techniques:

- KNN

Next:

- More extensions to linear & logistic regression
- Decision Trees & Random Forest

Beyond Linear Regression and Logistic Regression

- Nonlinear models with 1 predictor: $Y = f(X)$
  - The basis function approach
  - Regression Splines
  - Smoothing Splines
  - Local Regression (not covered)
- Nonlinear models with p predictors: $Y = f(X_1, X_2, \ldots, X_p)$
  - Generalized Additive Models (GAMs)

The Basis Function Approach (ISLR 7.1–7.3)

Linear regression: $Y \approx \beta_0 + \beta_1 X$

Logistic regression: $\log\left(\frac{\Pr(Y=1|X)}{1-\Pr(Y=1|X)}\right) \approx \beta_0 + \beta_1 X$

Adding high order terms:

$$Y \quad \text{or} \quad \text{log-odds}(Y) \approx \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots$$

Logarithmic terms:

$$\cdots\cdots \approx \beta_0 + \beta_1 \log(X)$$

More generally:

$$Y \quad \text{or} \quad \text{log-odds}(Y) \approx \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdots + \beta_K b_K(X)$$

- $b_1(\cdot), \ldots, b_K(\cdot)$: basis functions (pre-specified)

Polynomial Basis Functions

$$Y \quad \text{or} \quad \text{log-odds}(Y) \approx \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdots + \beta_K b_K(X)$$

Polynomial functions:

$$b_j(x) = x^j, \quad j = 1, \ldots, K$$

This leads to a polynomial model

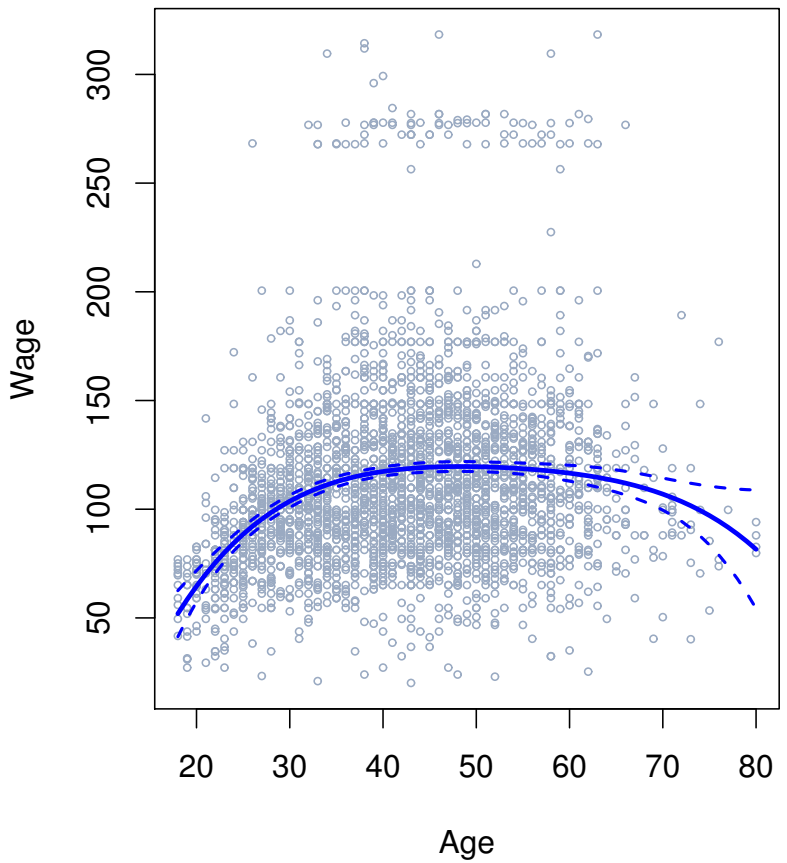
$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_K X^K$$

Example: Wage Dataset

$y_i =$ wage of individual i $\qquad$ $x_i =$ age of individual i

Regression with polynomial basis functions up to degree 4:

$$\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4$$



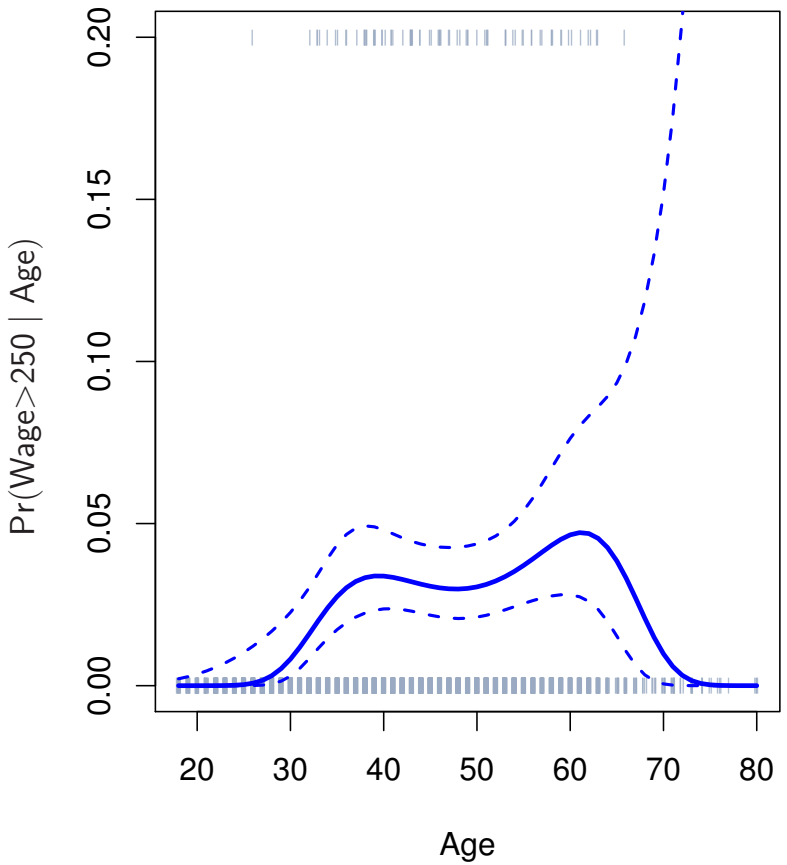
- Dotted lines: 95% confidence intervals of $\hat{y}_i$

Example: Wage Dataset

$y_i =$ wage of individual i $\qquad$ $x_i =$ age of individual i

Classification with polynomial basis functions up to degree 4:

$$\log\left[\frac{\hat{\Pr}(y_i > 250)}{1 - \hat{\Pr}(y_i > 250)}\right] = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4$$



Step Basis Functions

$$Y \quad \text{or} \quad \text{log-odds}(Y) \approx \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdots + \beta_K b_K(X)$$

Step functions: Given knots $c_1, c_2, \ldots, c_K$

$$\begin{aligned} b_1(x) &= C_1(x) \triangleq I(c_1 \leq x < c_2) \\ b_2(x) &= C_2(x) \triangleq I(c_2 \leq x < c_3) \\ &\vdots \\ b_{K-1}(x) &= C_{K-1}(x) \triangleq I(c_{K-1} \leq x < c_K) \\ b_K(x) &= C_K(x) \triangleq I(c_K \leq x) \end{aligned}$$

This leads to a piecewise-constant model

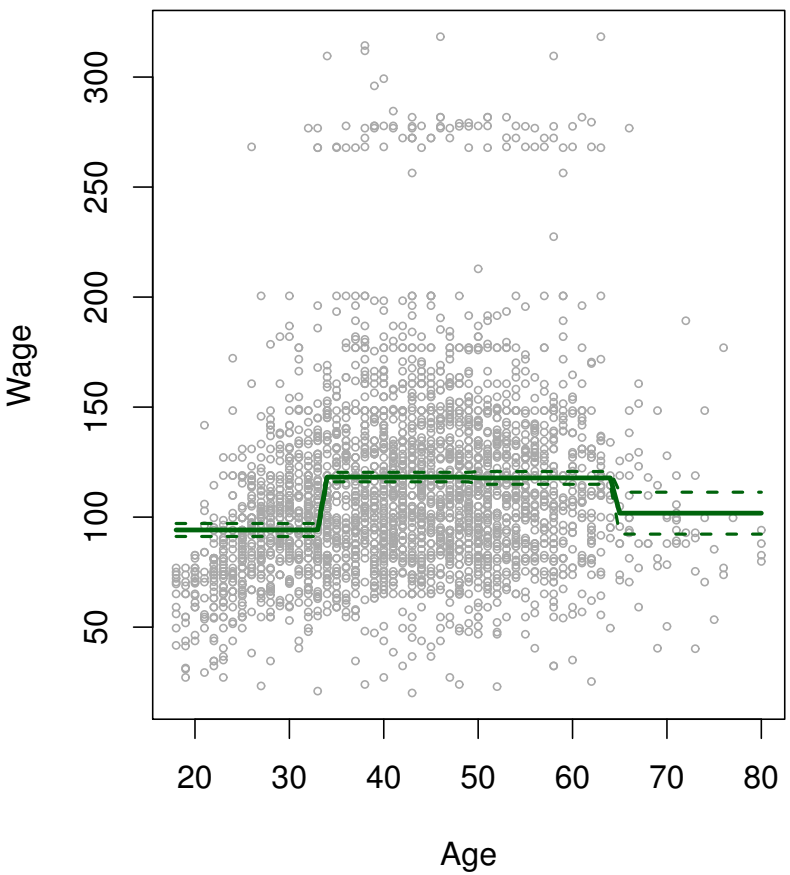
- knots need to be pre-specified (often not clear how to do so)

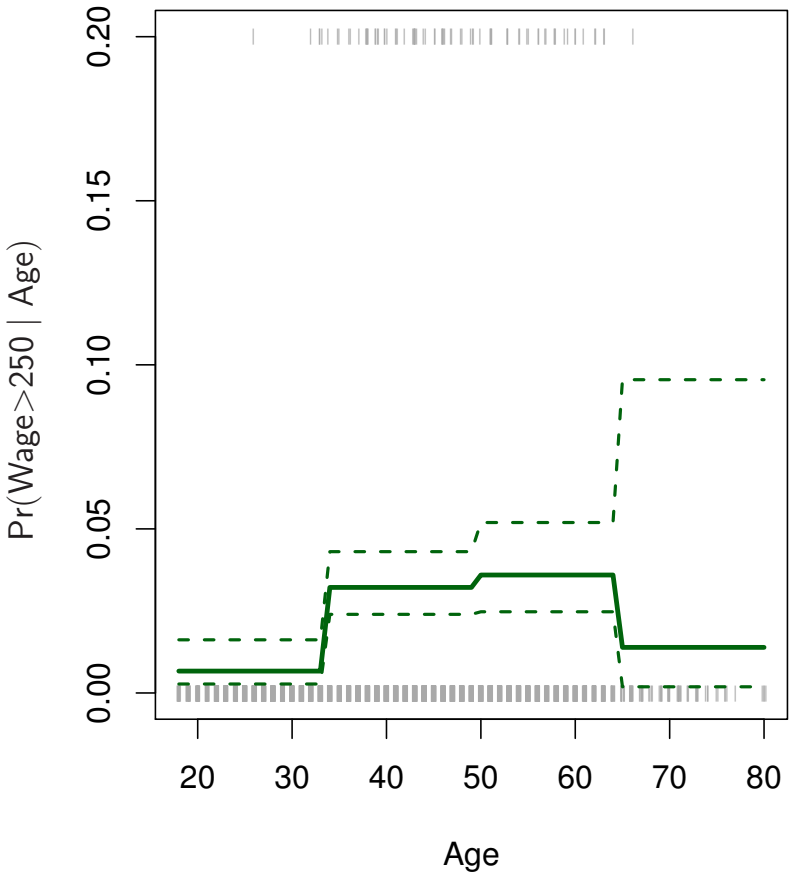
Example: Wage Dataset

y_i = wage of individual i $\qquad$ x_i = age of individual i

Use step basis functions with 2 knots:

$$y_i \text{ or } \log\left[\hat{\Pr}(y_i > 250)/(1 - \hat{\Pr}(y_i > 250))\right] \approx \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i)$$





The Basis Function Approach

$$Y \quad \text{or} \quad \text{log-odds}(Y) \approx \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdots + \beta_K b_K(X)$$

Fitted by least squares

Can use all the tools from linear regression:

- Standard errors & confidence intervals for $\hat{\beta}_j$
- p-values for each $\hat{\beta}_j$
- p-values for the entire model

Other choices of basis functions:

- $b_1(x) = \sqrt{x}$
- $b_1(x) = \log(x)$
- Based on wavelets or Fourier series (not covered)
- Regression Splines (next)

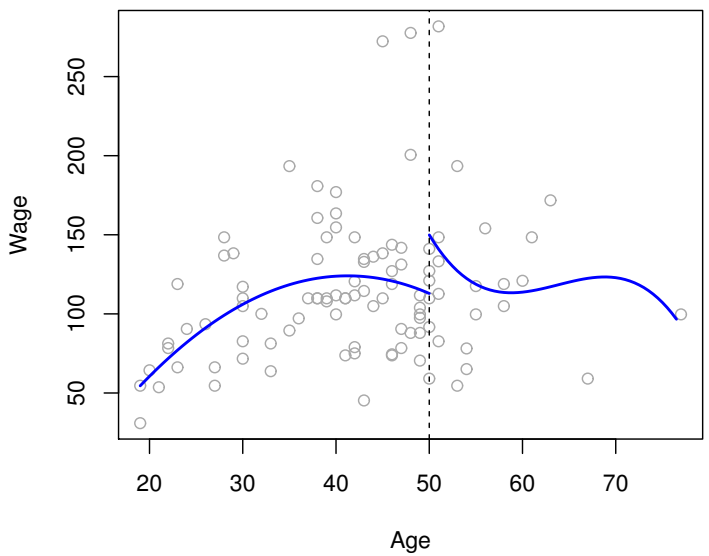
Regression Splines (ISLR 7.4)

Using step functions, we fit a piecewise constant model

$$Y \approx \beta_0 + \beta_1 C_1(X) = \begin{cases} \beta_0 & \text{if } X < c \\ \beta_0 + \beta_1 & \text{if } X \geq c \end{cases}$$

More generally, we can fit a piecewise polynomial model

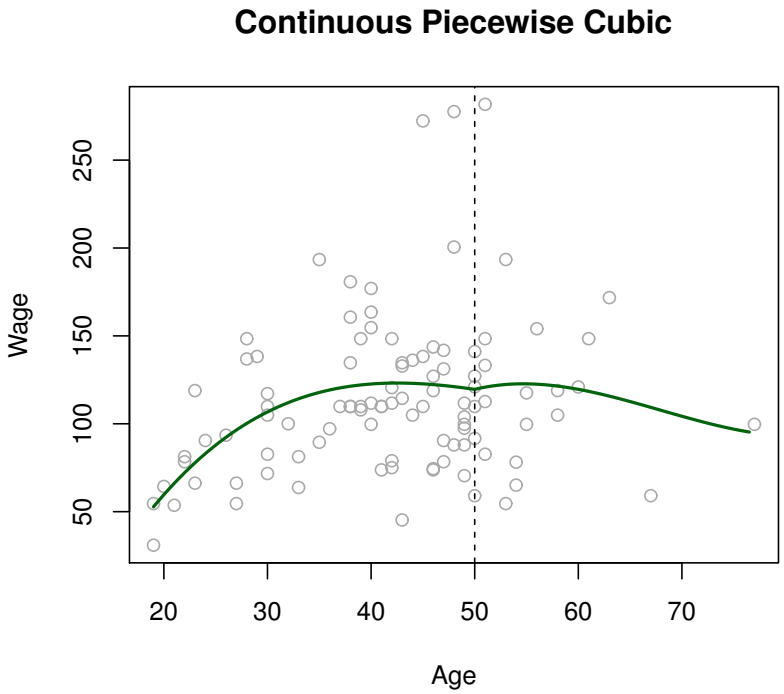
$$Y \approx \begin{cases} \beta_{01} + \beta_{11} X + \beta_{21} X^2 + \beta_{31} X^3 & \text{if } X < c \\ \beta_{02} + \beta_{12} X + \beta_{22} X^2 + \beta_{32} X^3 & \text{if } X \geq c \end{cases}$$

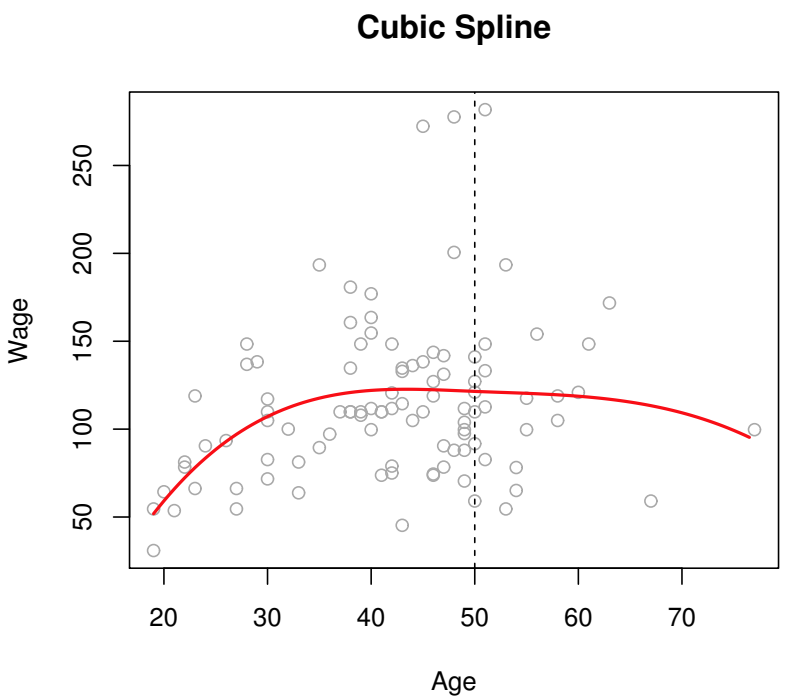


- 8 degrees of freedom (too flexible)

Regression Splines

Regression splines:
Piecewise polynomial models that are continuous and smooth at the knots (smoothness = continuity of derivatives)





Most popular: Cubic splines

- Continuous piecewise cubic models with continuous first two derivatives
- K knots: $K + 4$ degrees of freedom (instead of $4K + 4$)
- Reduce flexibility/variance; increase bias

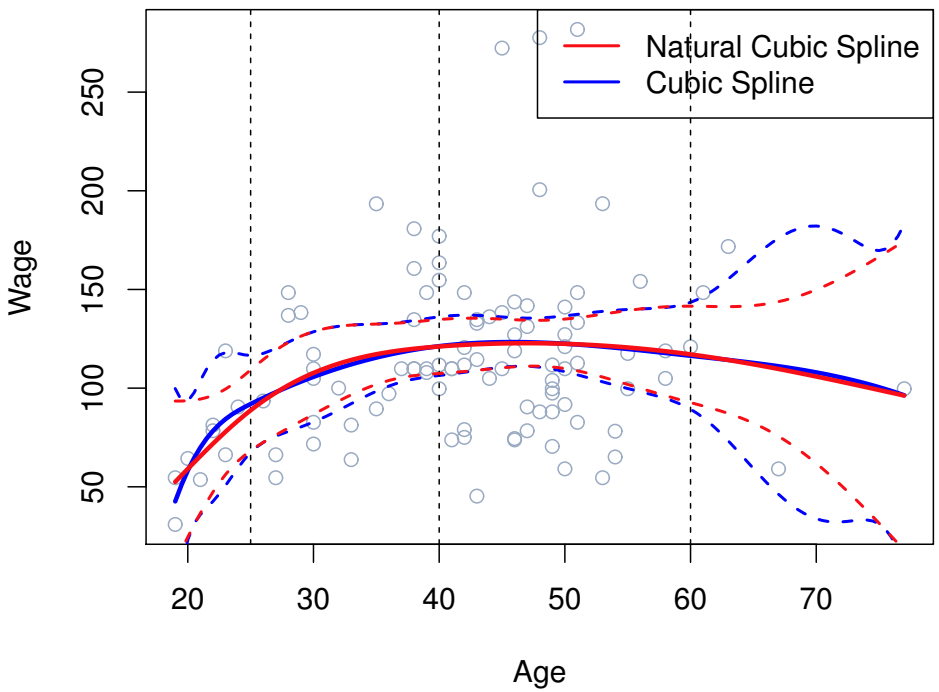
Cubic Splines

- A cubic splines with K knots ($K + 4$ DF) can be written as

$$Y \approx \beta_0 + \beta_1 b_1(X) + \cdots + \beta_{K+3} b_{K+3}(X)$$

with appropriate basis functions $b_j(\cdot)$ (cf. ISLR 7.4.3)

- So can be fitted using least squares



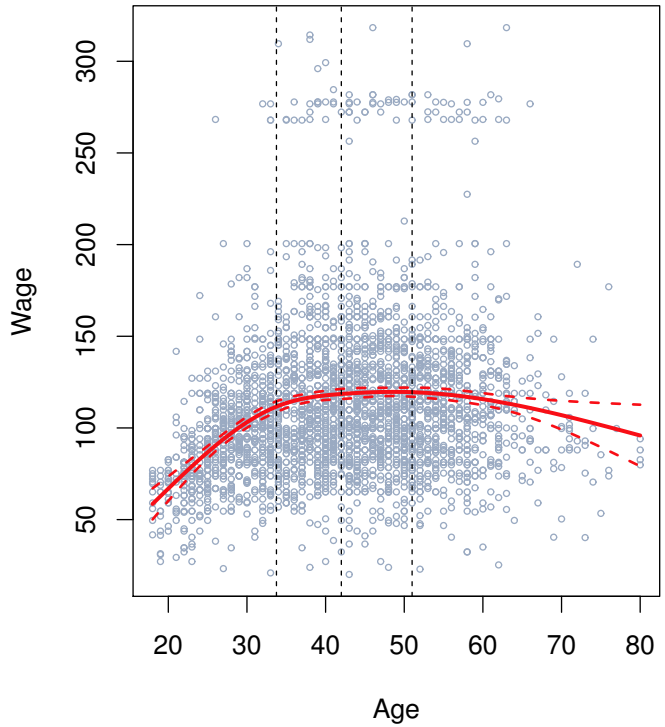
- Natural cubic splines: linear at the boundary

(further reduce df/flexibility/variance)

Cubic Splines: Choosing the Knots

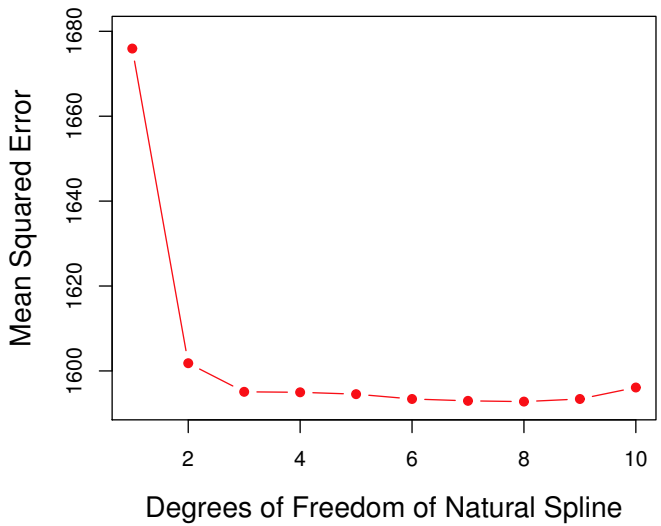
Locations of knots:

- Placed at uniform quantiles of data



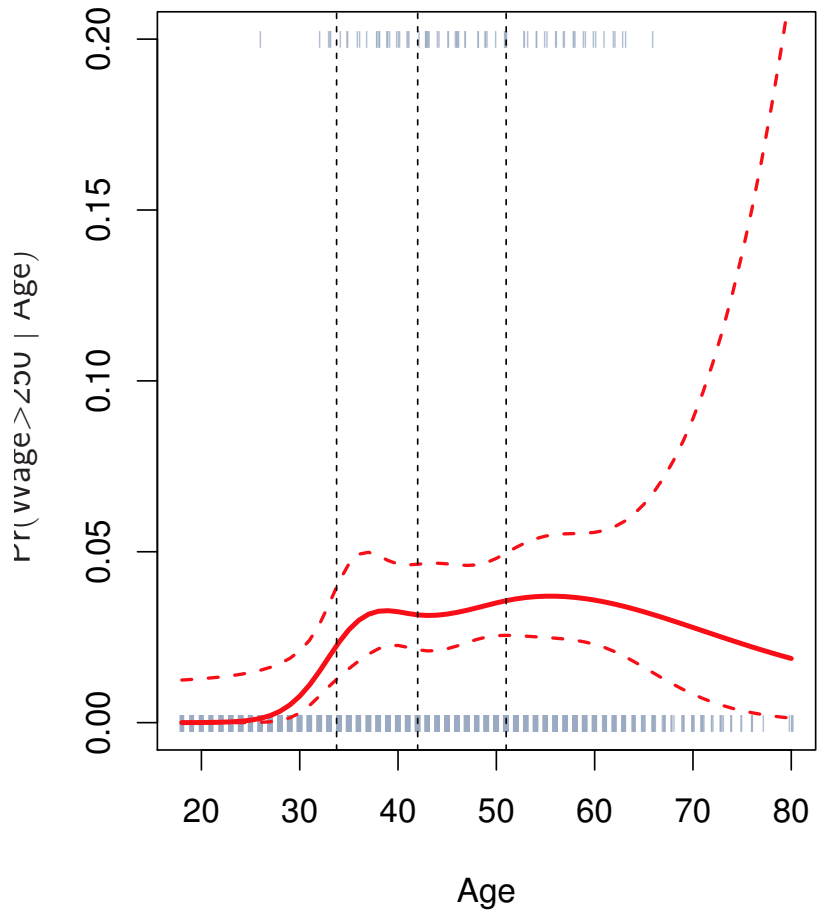
Number of knots:

- Equivalent to choosing degrees of freedom
- Choose the best-looking curve, or...
- By cross-validation



Cubic Splines

Apply to classification (logistic regression) as well



Polynomial Regression vs. Cubic Splines

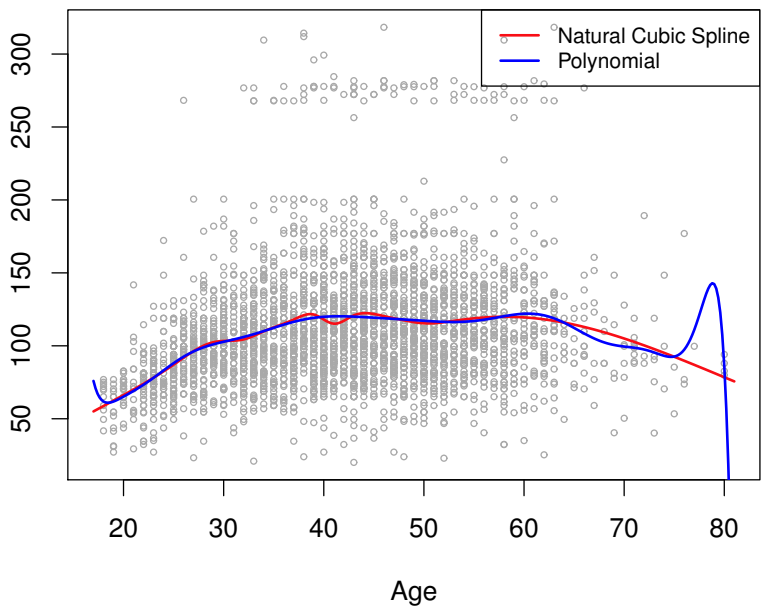
Polynomial regression (the basis function approach with polynomial basis)

$$Y \approx \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_K X^K$$

- Flexibility/DF determined by degree of polynomials K

Cubic splines

- Flexibility/DF determined by number of knots K



Same degrees of freedom (=15)

Cubic splines often more stable (esp. at the boundaries)

Smoothing Splines (ISLR 7.5)

Recall:
(Cubic) Regression splines:

- Specify knots (or DF)
- Cubic polynomials between knots
- Require smoothness at knots
- Fitting: convert to a basis function model and solved by LS

Smoothing splines: Another way of fitting a smooth curve $g(\cdot)$

- Specify tuning parameter λ
- Find curve as the solution to the optimization problem

$$\min_g \underbrace{\sum_{i=1}^{n}(y_i - g(x_i))^2}_{\text{Loss (RSS)}} + \underbrace{\lambda \int g''(t)^2 dt}_{\text{Regularization}}$$

Smoothing Splines

$$\min_g \underbrace{\sum_{i=1}^{n}(y_i - g(x_i))^2}_{\text{Loss (RSS)}} + \underbrace{\lambda \int g''(t)^2 dt}_{\text{Regularization}}$$

- Loss term: encourage $g(\cdot)$ to fit data well
- Regularization: encourage smoothness
- $g''(t)$: second derivative
- Small $g''(t)$: less wiggly near t
- Larger $\lambda \Rightarrow$ Smaller $g''(t) \Rightarrow g(\cdot)$ more smooth

The optimal solution

- Can show: the optimal $g(\cdot)$ is a natural cubic spline
- with knots at $x_1, x_2, \cdots, x_n$
- n knots , but less than $n+4$ DF (b/c of λ)

Smoothing Splines: Choosing λ

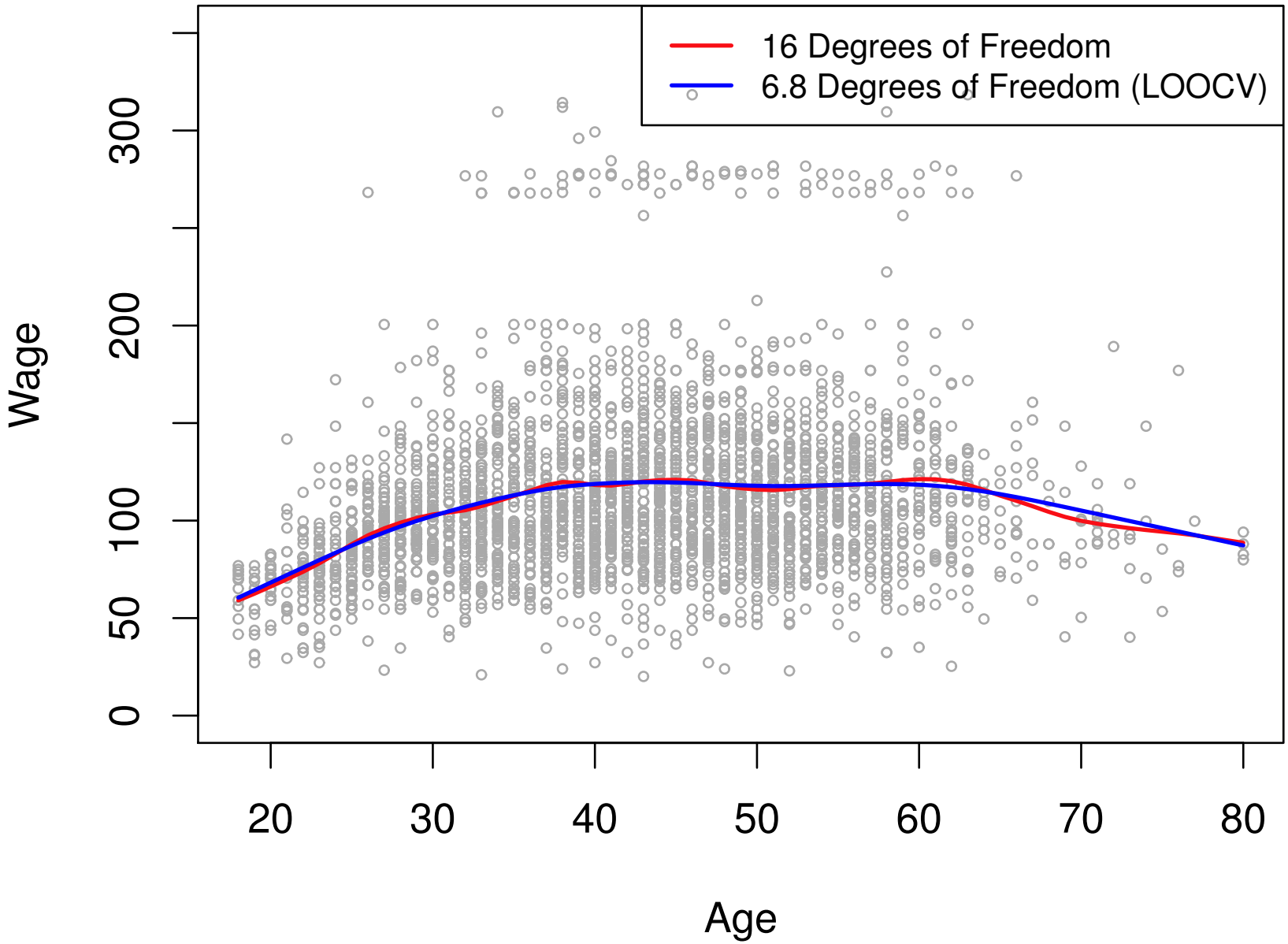
$$\min_g \quad \sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

(Recall: In regression splines, flexibility determined by # knots K, or DF $K + 4$)

For smoothing splines:

- Flexibility determined by λ
- Corresponding to an effective degree of freedom, df_λ
- Closed form expression for df_λ (cf. ISLR 279)
- Choose λ (or df_λ) by CV
- LOOCV can be done very efficiently

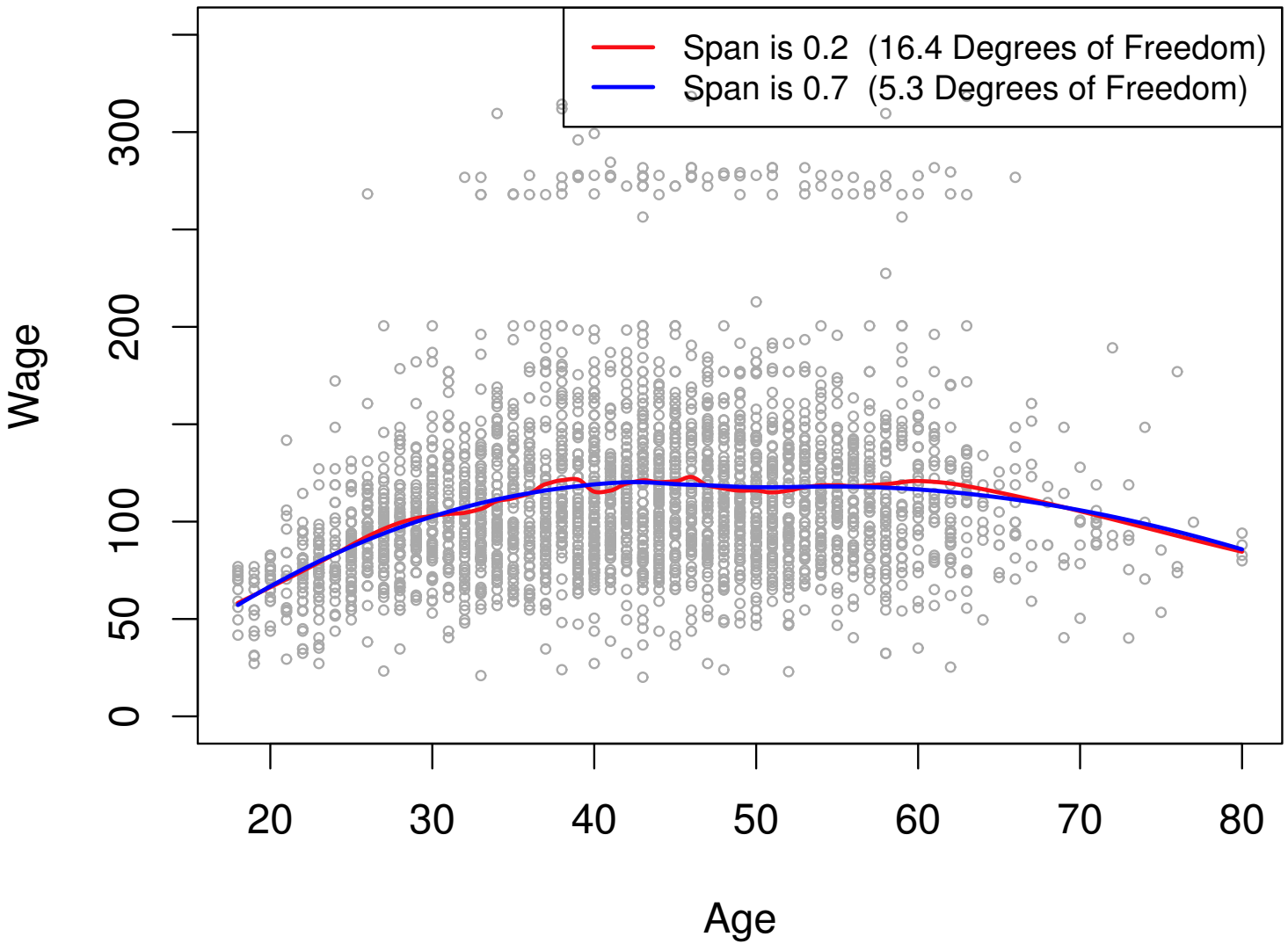
Smoothing Splines: Choosing λ



Local Regression (Not covered; ISLR 7.6)

A third way of fitting smooth curves

- Flexibility determined by a tuning parameter *s* (span)
- Corresponding to some effective DF



Mini Summary

1 predictor: $Y = f(X)$

- Basis function approach: $f(X) = \sum_j \beta_j b_j(X)$
- Regression Splines: $f(X) =$ piecewise polynomials joint smoothly
- Smoothing Splines: $f(X) =$ solution to $f''(\cdot)$-regularized least squares
- Local Regression (not covered)

p predictors: $Y = f(X_1, X_2, \ldots, X_p)$

- Generalized Additive Models (GAMs)

Generalized Additive Models (ISLR 7.7)

Recall: Multiple linear regression

$$Y \approx \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

Generalized Additive Model: Maintains only additivity

$$Y \approx \beta_0 + f_1(X_1) + \cdots + f_p(X_p)$$

- $f_j(\cdot)$: Any of the univariate nonlinear functions we just learned
- E.g. polynomials, linear combination of basis functions, cubic/smoothing splines
- Build multivariate nonlinear models by adding up univariate ones

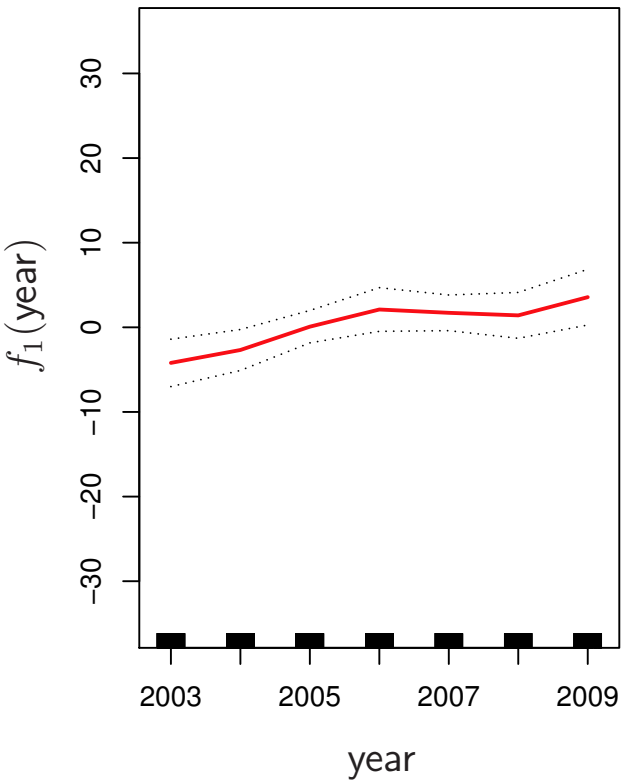
Example: Wage Dataset

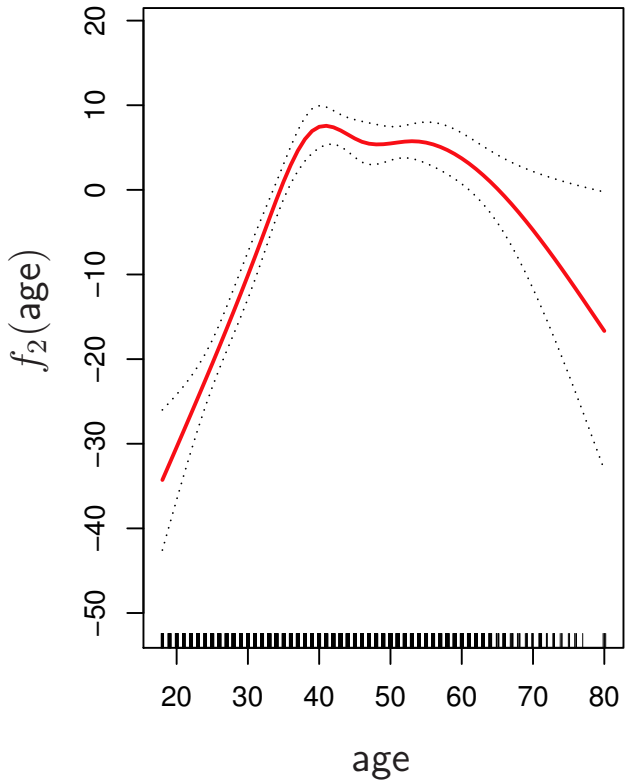
Fit a GAM of the form

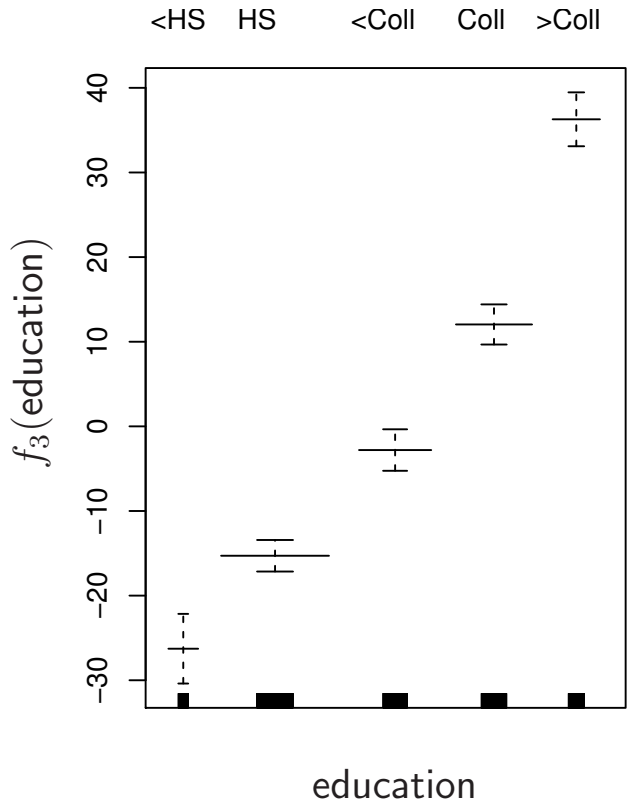
$$\text{wage} \approx \beta_0 + f_1(\text{year}) + f_2(\text{age}) + f_3(\text{education})$$

where

- $f_1(\cdot)$, $f_2(\cdot)$: natural cubic splines
- education: categorical w/ 5 levels `<HS, HS, <Coll, Coll, >Coll`
- $f_3(\cdot) =$ a different value for each level of education
  - i.e., encode education w/ 4 four dummy variables and fit a usual linear model


f_1(year)
30
20
10
0
−10
−20
−30
2003
2005
2007
2009
year


f_2(age)
20
10
0
−10
−20
−30
−40
−50
20
30
40
50
60
70
80
age


<HS
HS
<Coll
Coll
>Coll
f_3(education)
40
30
20
10
0
−10
−20
−30
education

GAMs for Classification

Recall: Logistic regression

$$\log\left(\frac{\Pr(Y=1|X)}{1-\Pr(Y=1|X)}\right) \approx \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

Logistic regression GAM:

$$\log\left(\frac{\Pr(Y=1|X)}{1-\Pr(Y=1|X)}\right) \approx \beta_0 + f_1(X_1) + \cdots + f_p(X_p)$$

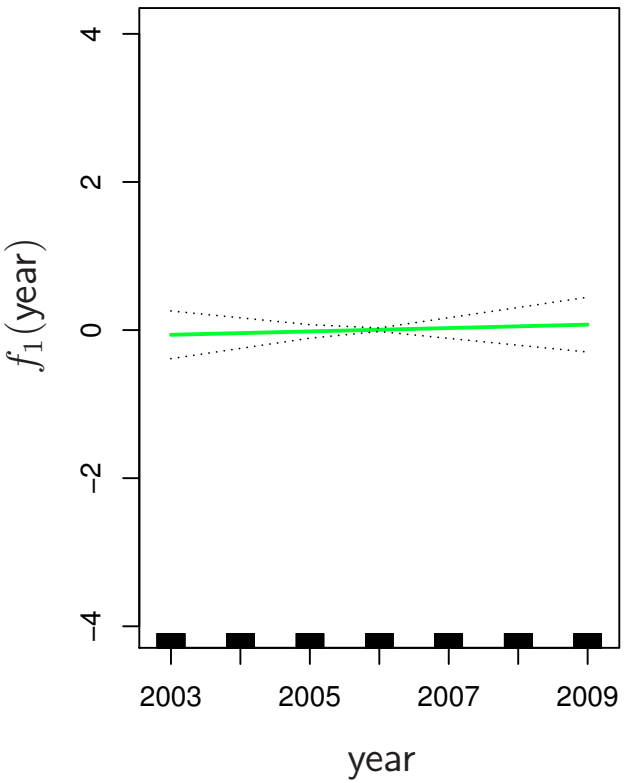
Example: Wage Dataset

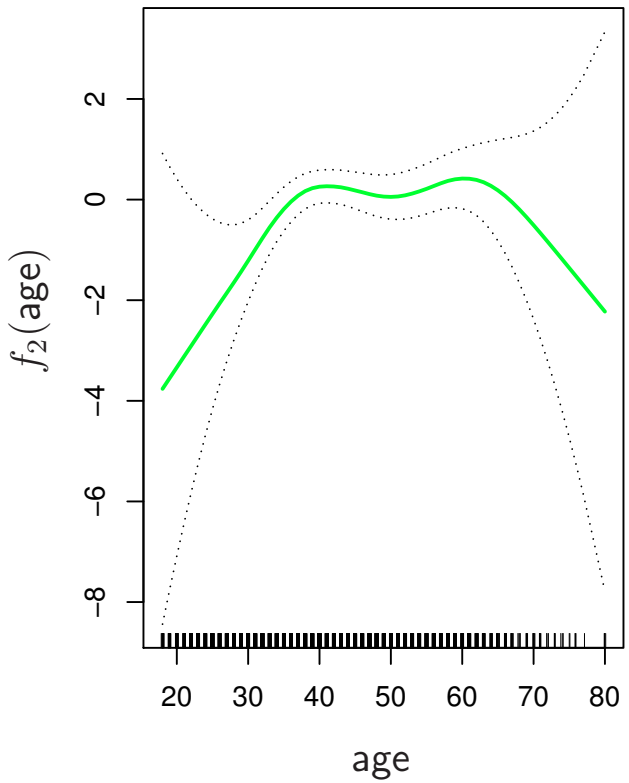
Fit a GAM of the form

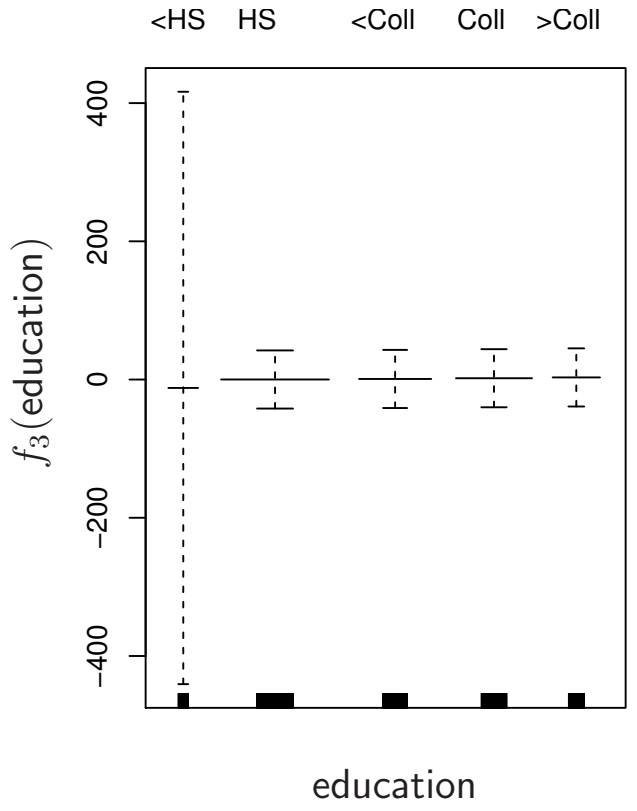
$$\log\left(\frac{\Pr(\text{wage} > 250)}{\Pr(\text{wage} \leq 250)}\right) \approx \beta_0 + \beta_1 \times \text{year} + f_2(\text{age}) + f_3(\text{education})$$

where

- $f_2(\cdot)$: smoothing splines with $df = 5$
- $f_3(\cdot)$ constant for each level of education


f_1(year)
4
2
0
−2
−4
2003
2005
2007
2009
year


f_2(age)
2
0
−2
−4
−6
−8
20
30
40
50
60
70
80
age


<HS
HS
<Coll
Coll
>Coll
f_3(education)
400
200
0
−200
−400
education

GAMs: Pros and Cons

$$Y \quad \text{or} \quad \text{log-odds}(Y) \approx \beta_0 + f_1(X_1) + \cdots + f_p(X_p)$$

- Combine simple univariate nonlinear models $f_j(\cdot)$ to build p-variate models
- Flexible choices for each $f_j(\cdot)$
- (Natural) Cubic Spline is a popular choice
- Control flexibility by specifying degrees-of-freedom

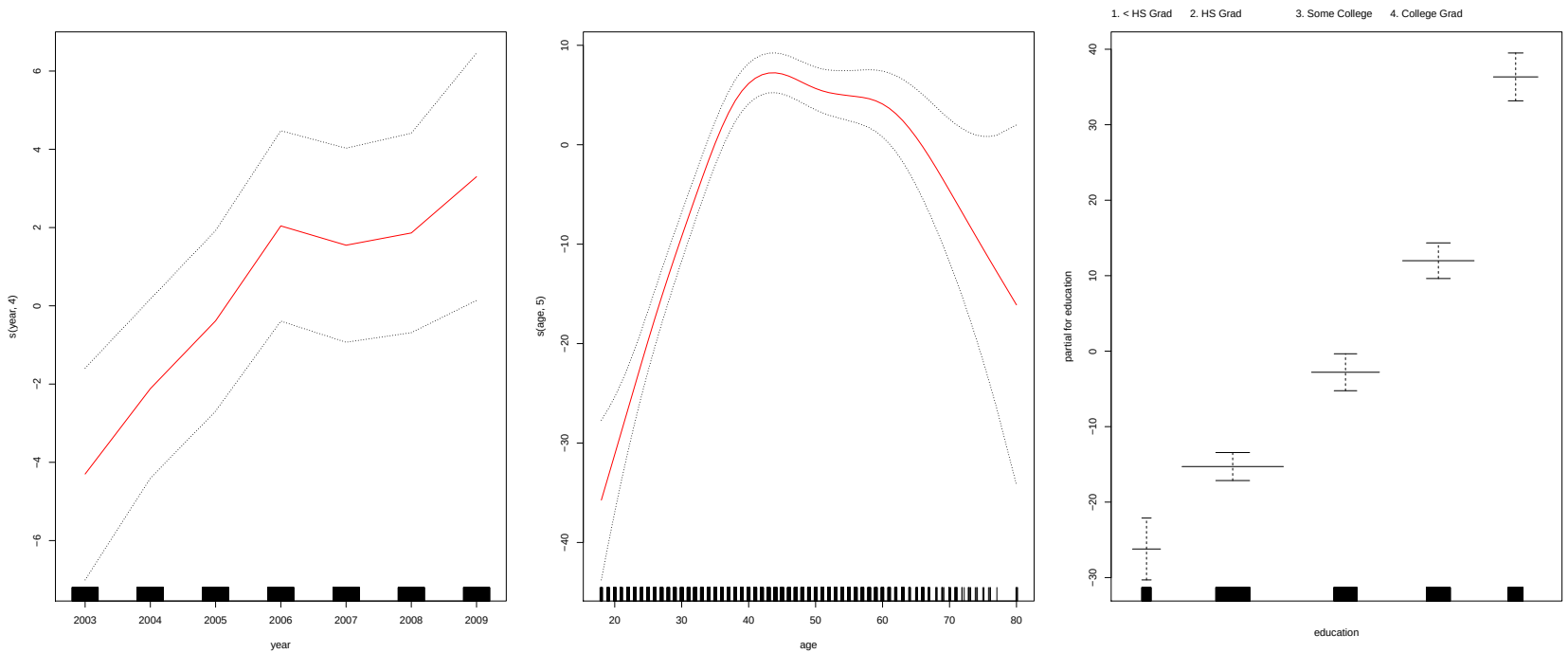
- Interaction/synergy effects b/w predictors not captured

Nonlinear Modeling in R (ISLR 7.8)

- GAM with smoothing splines

```
> library(gam)
> gam2 = gam(wage~s(year,4)+s(age,5)+education, data=Wage)

> plot(gam2, se=TRUE, col="red")
```

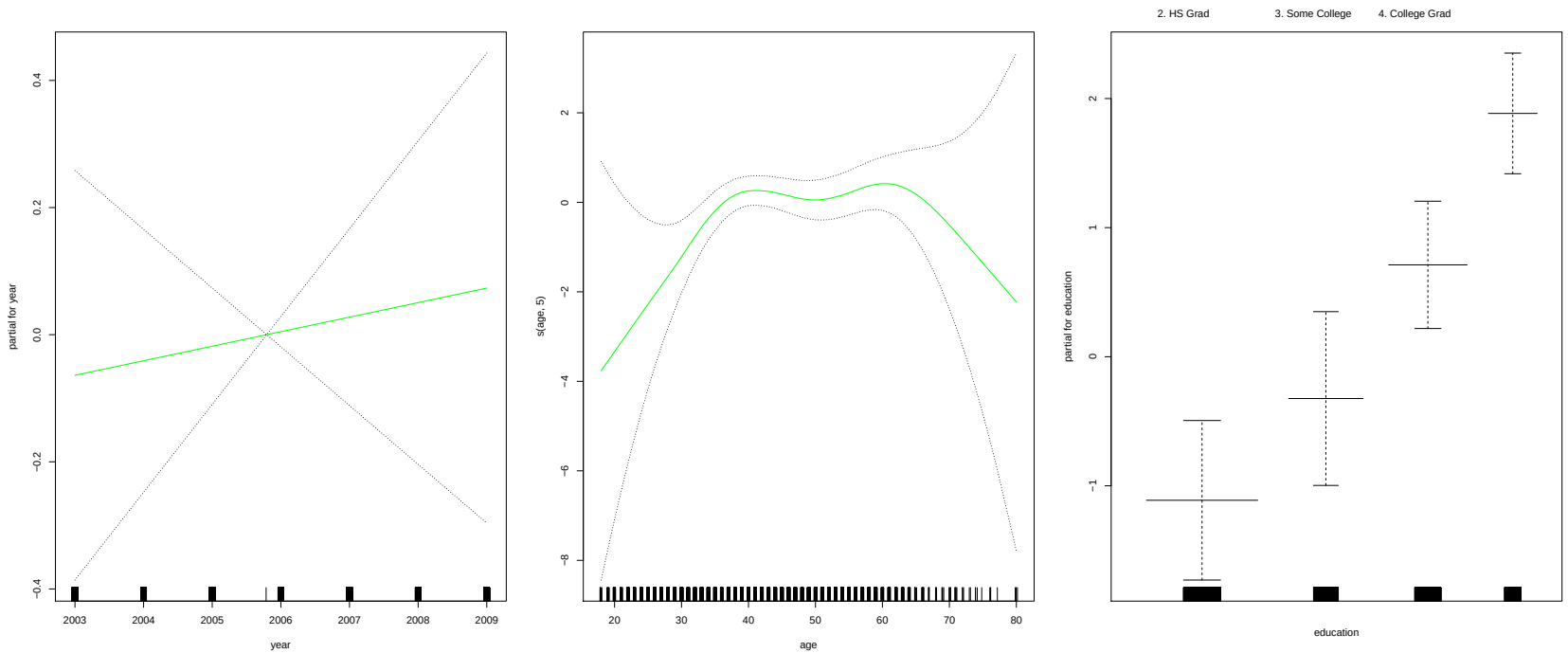


Nonlinear Modeling in R (ISLR 7.8)

- Logistic regression GAM with smoothing splines

Excluding observations with less than a high school education

```
> library(gam)
> gam.lr = gam(I(wage>250)~year+s(age,5)+education, family=
+ binomial, data=Wage, subset=(education!="1. < HS Grad"))
> plot(gam.lr, se=TRUE, col="green")
```



Nonlinear Modeling Summary

1 predictor: $Y = f(X)$

- Basis function approach: $f(X) = \sum_j \beta_j b_j(X)$
- Regression Splines: $f(X) =$ piecewise polynomials joint smoothly
- Smoothing Splines: $f(X) =$ solution to $f''(\cdot)$-regularized least squares
- Local Regression

p predictors: $Y = f(X_1, X_2, \ldots, X_p)$

- Generalized Additive Models (GAMs)

$$Y \quad \text{or} \quad \text{log-odds}(Y) \approx \beta_0 + f_1(X_1) + \cdots + f_p(X_p)$$

where $f_j(\cdot)$ is a polynomial, step function, cubic/smoothing spline, local regression, ...

Decision Trees and Random Forests

Yudong Chen
School of ORIE, Cornell University
ORIE 4740 Lec 20-21

Recap: Linear vs. Nonlinear

Linear techniques:

- Linear and logistic regression
- k-means; PCA

Simple extensions of linear techniques:

- Adding high-order and interaction terms
- Converting to dummy variables

Nonlinear techniques:

- KNN
- Basis functions, splines and GAM

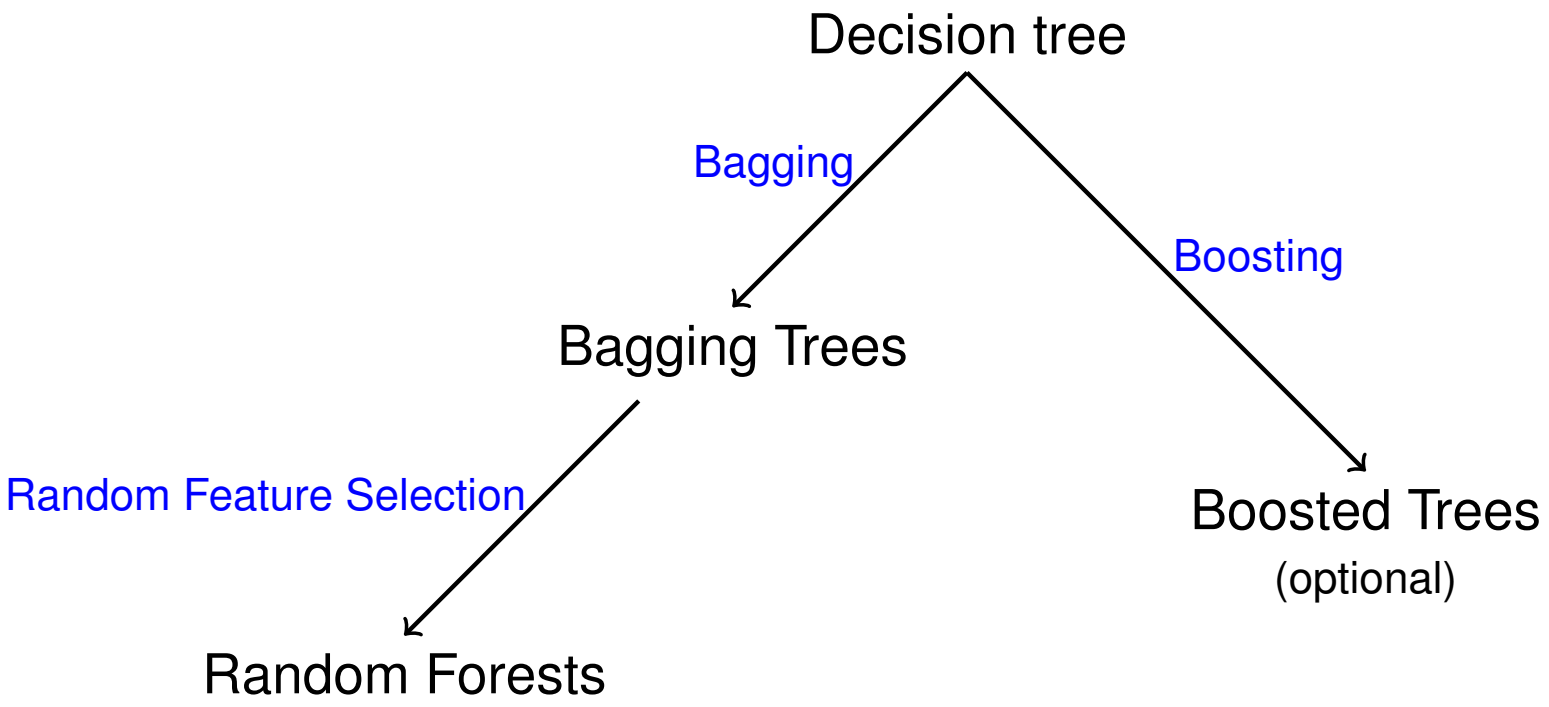
Next:

- Decision Trees & Random Forests

Key issues:

- What determines model flexibility (#variables, λ, k, DoF)
- How to choose it (by CV, p-values, R^2, ANOVA.....)

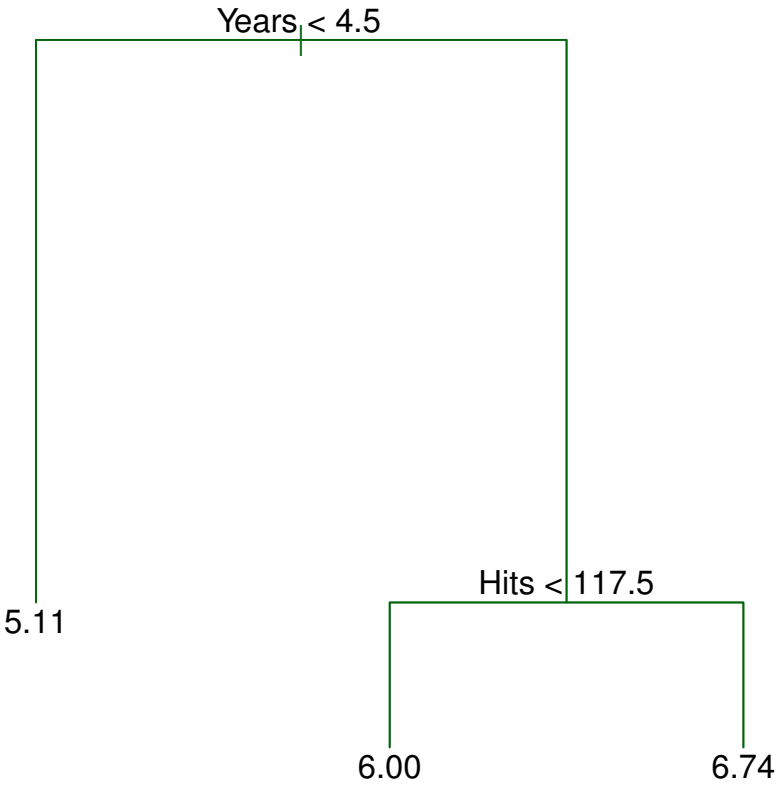
Tree-based Methods

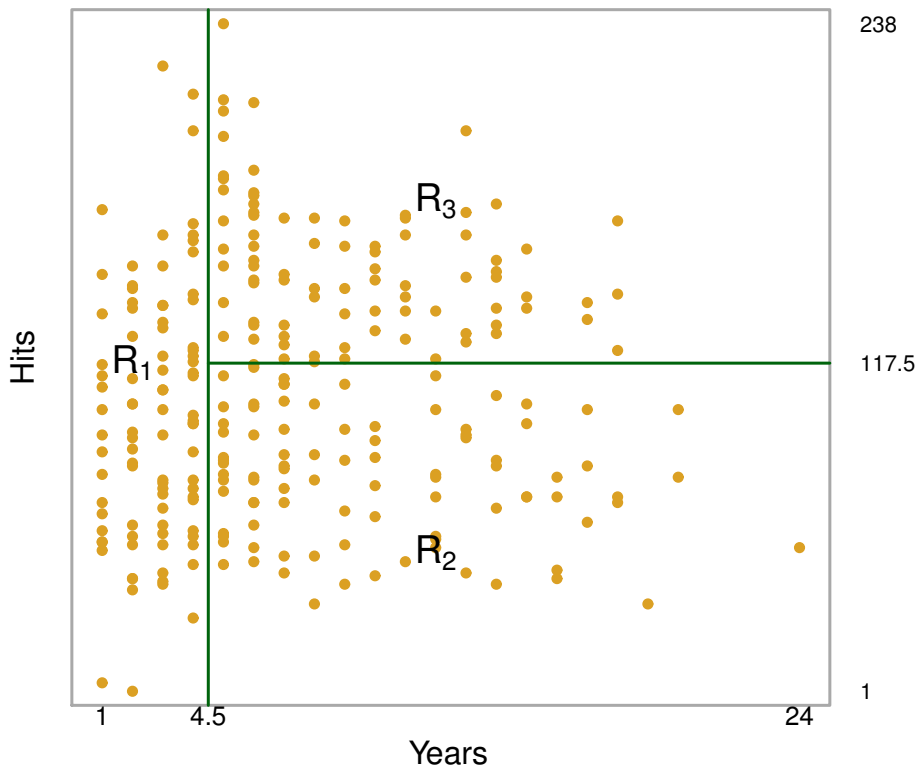


- Popular for classification, but works for regression as well
- The model is easy to explain, but the fitting procedure is harder

Regression Trees (ISLR 8.1.1)

- Decision trees for regression
- Leaf, internal node, branch, split

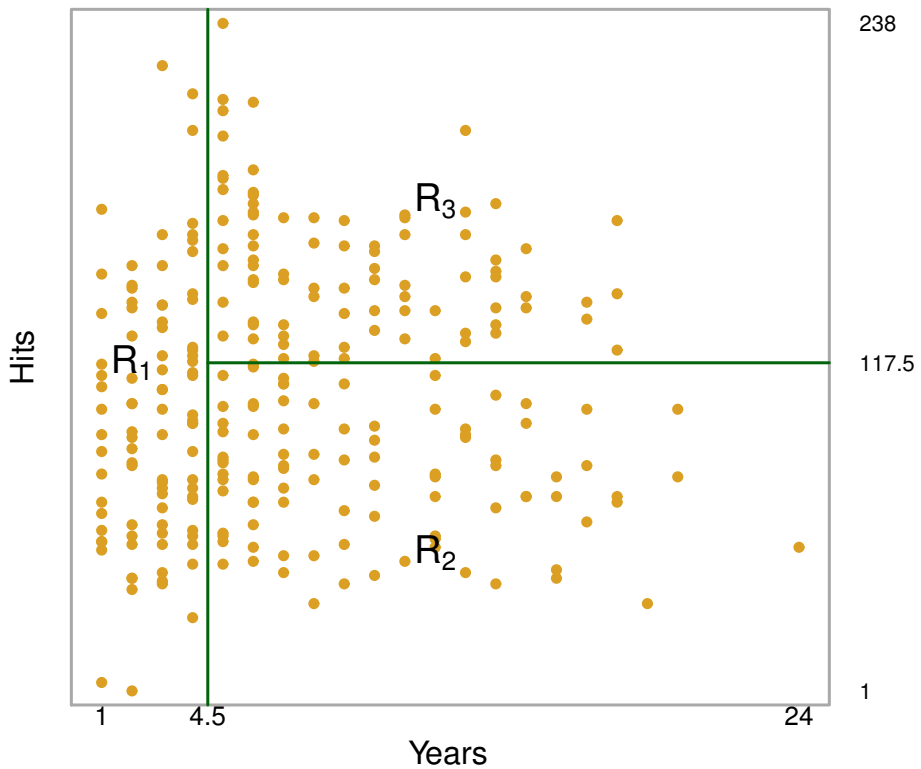




Regression Tree: Prediction

- Predictor space partitioned into J non-overlapping regions $R_1, R_2, \ldots, R_J$
- If an observation $(x_1, \ldots, x_p)$ falls into R_j:

$$\hat{y} = \hat{y}_{R_j} \triangleq \text{mean of } y_i\text{'s in of observations in } R_j$$

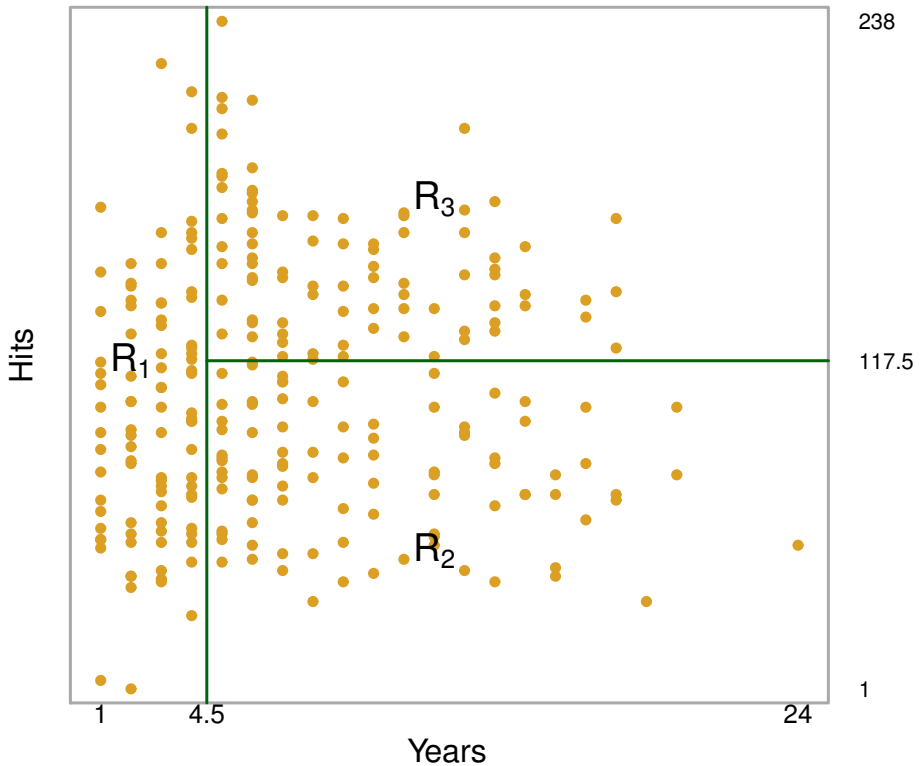


Regression Tree: Fitting

- $R_1, R_2, \ldots, R_J$ are rectangles/boxes
- Ideally, want to minimize

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- Instead, we fit a tree greedily
  - First grow a large tree
  - Then prune the tree



Growing a Regression Tree

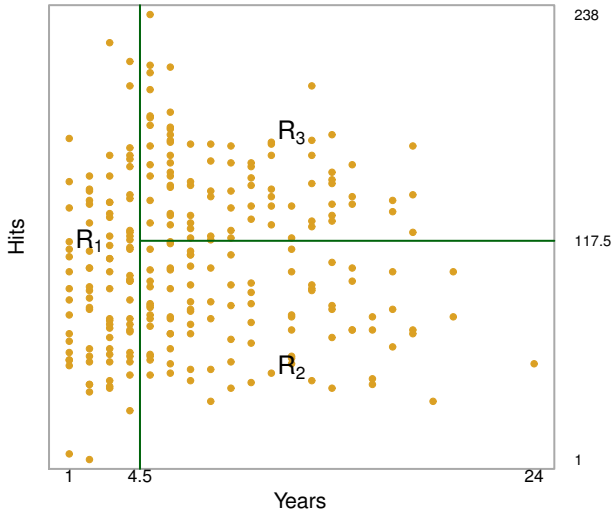
Growing: split the predictor space greedily

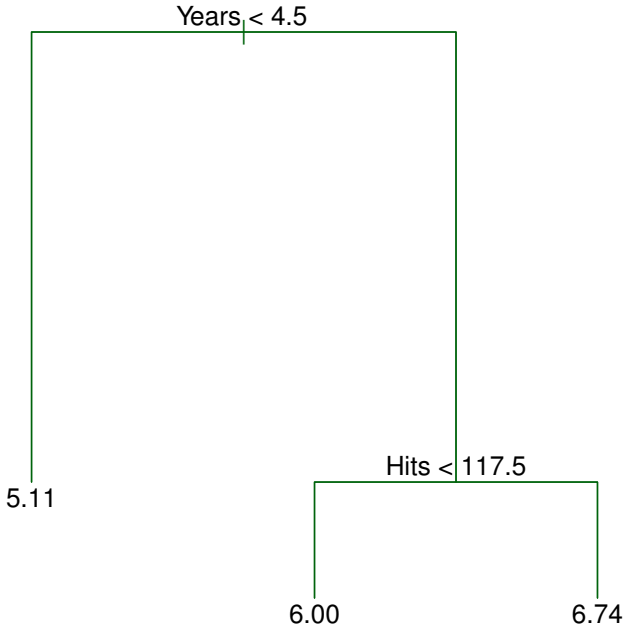
Recursive binary splitting:

1. Find a predictor X_j and a cutpoint s that gives the best RSS by splitting the predictor space into

$$\{X|X_j < s\} \text{ and } \{X|X_j \geq s\}$$

2. Repeat: find the best region to split, and the best way to split it





Pruning a Regression Tree

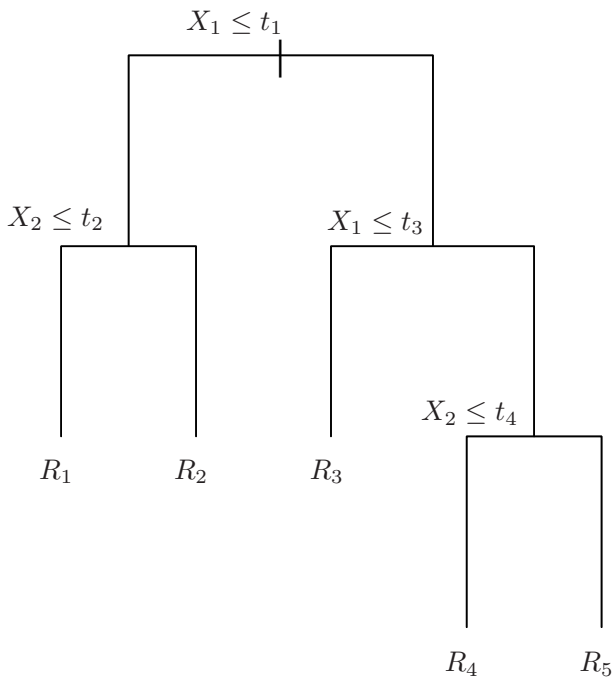
Pruning: Find a subtree that minimizes the CV error

"Cost complexity pruning":

1. For each α, find the subtree T that minimizes

$$\text{RSS}(T) + \alpha|T|$$

   where $|T| = \#\text{leaves} = \#\text{regions}$
2. As α increases, branches are cut off sequentially
3. Find the best α by CV

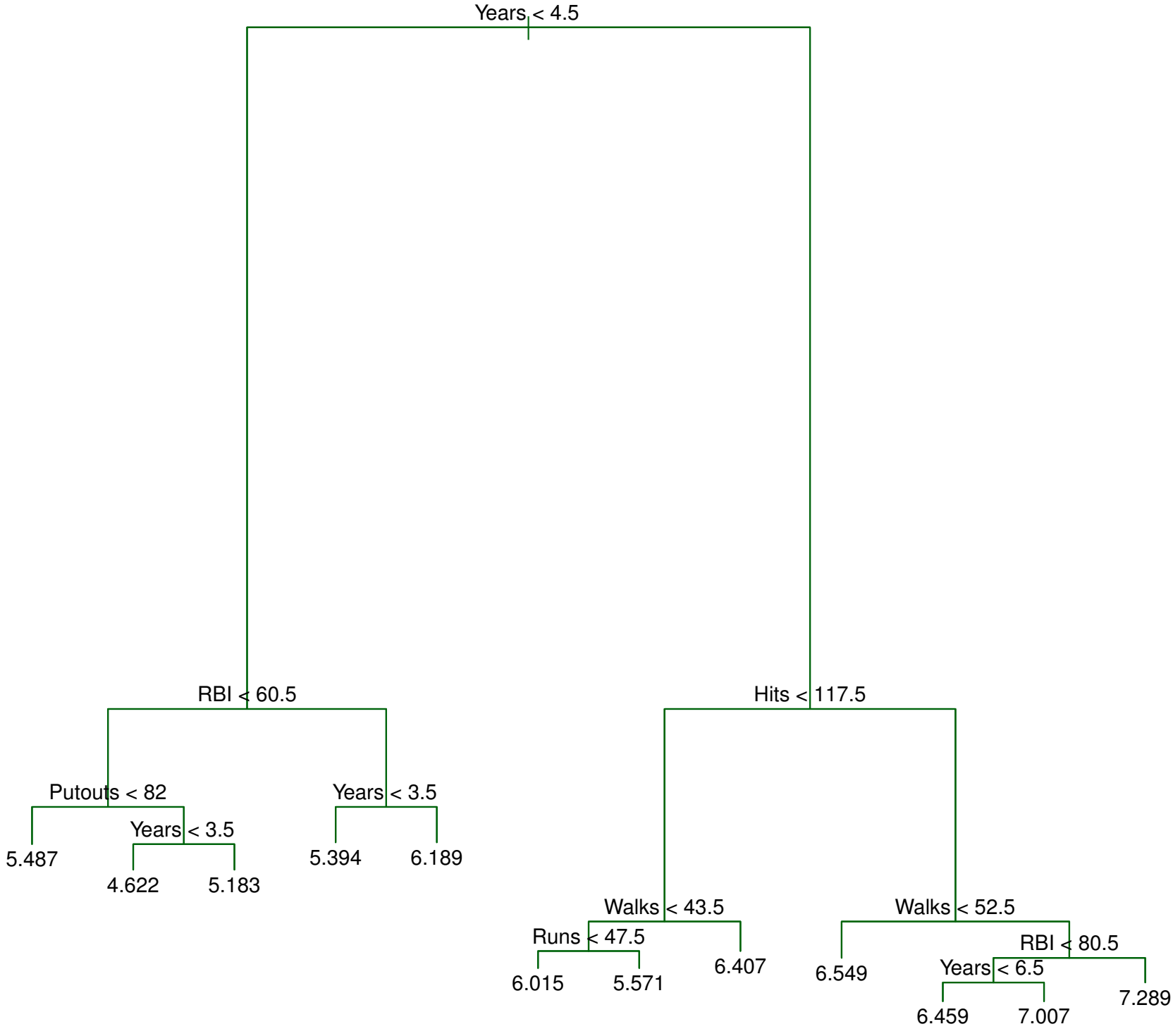


Regression Tree Fitting Algorithm

1. Grow a large tree by recursive binary splitting
   Stop when a leaf/region has too few observations

2. For each value of α, obtain a subtree by cost complexity pruning

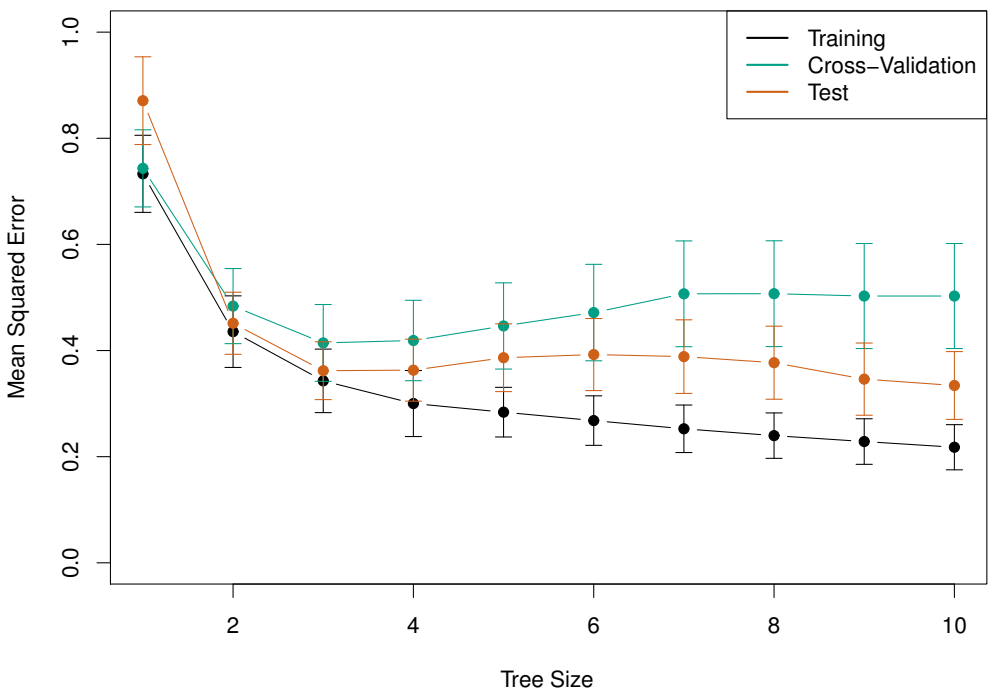
3. Compute the CV error of the subtree corresponding to each α.
   Pick and output the best subtree


Years < 4.5
RBI < 60.5
Hits < 117.5
Putouts < 82
Years < 3.5
Years < 3.5
Walks < 43.5
Walks < 52.5
Runs < 47.5
RBI < 80.5
Years < 6.5
5.487
4.622
5.183
5.394
6.189
6.015
5.571
6.407
6.549
6.459
7.007
7.289

Example: Hitters dataset

Response: Salary
Predictors: Year, Hits, RBI, Putouts, Walks, Runs

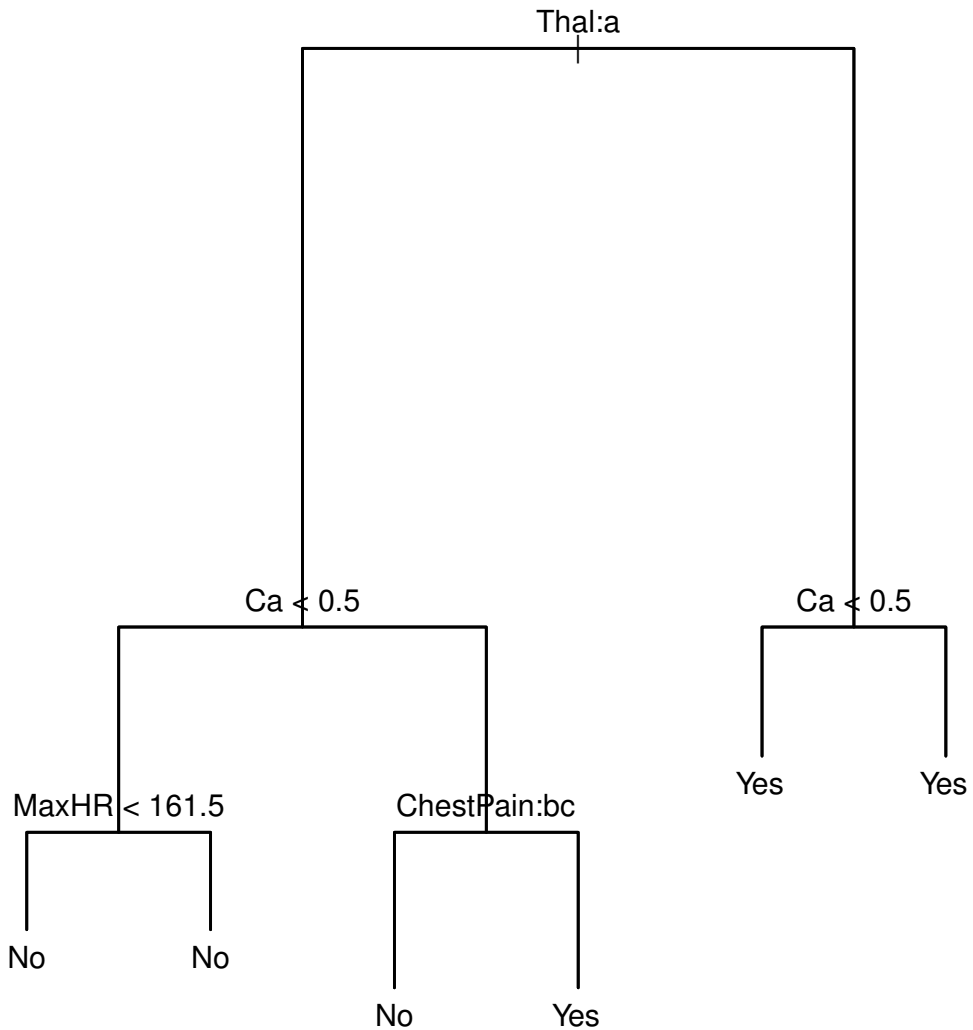
- Grow a large tree
- Prune the tree and select α by CV



- $\alpha \Rightarrow$ tree size = #leaves = #regions $\Rightarrow$ model flexibility

Classification Trees (ISLR 8.1.2)

Heart dataset
Response: Have heart disease or not
13 Predictors (numeric and categorical): Age, Sex, ChestPain, MaxHR, Thal, Ca . . .



Classification Tree: Prediction

- Predictor space partitioned into J non-overlapping regions $R_1, R_2, \ldots, R_J$
- If an observation $(x_1, \ldots, x_p)$ falls into R_j:

$$\hat{y} = \text{most common class of observations in } R_j$$

Classification Tree: Fitting

Algorithm:

1. Grow a large tree by recursive binary splitting
   Stop when a leaf/region has too few observations
2. For each value of α, obtain a subtree by cost complexity pruning
3. Compute the CV error of the subtree corresponding to each α.
   Pick and output the best subtree

- Need a different metric than RSS
- Natural choice: the classification error rate at each region R_m:

$$E = 1 - \max_k(\hat{p}_{mk})$$

- Another choice: the Gini index, measuring the *purity* of a region R_m:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

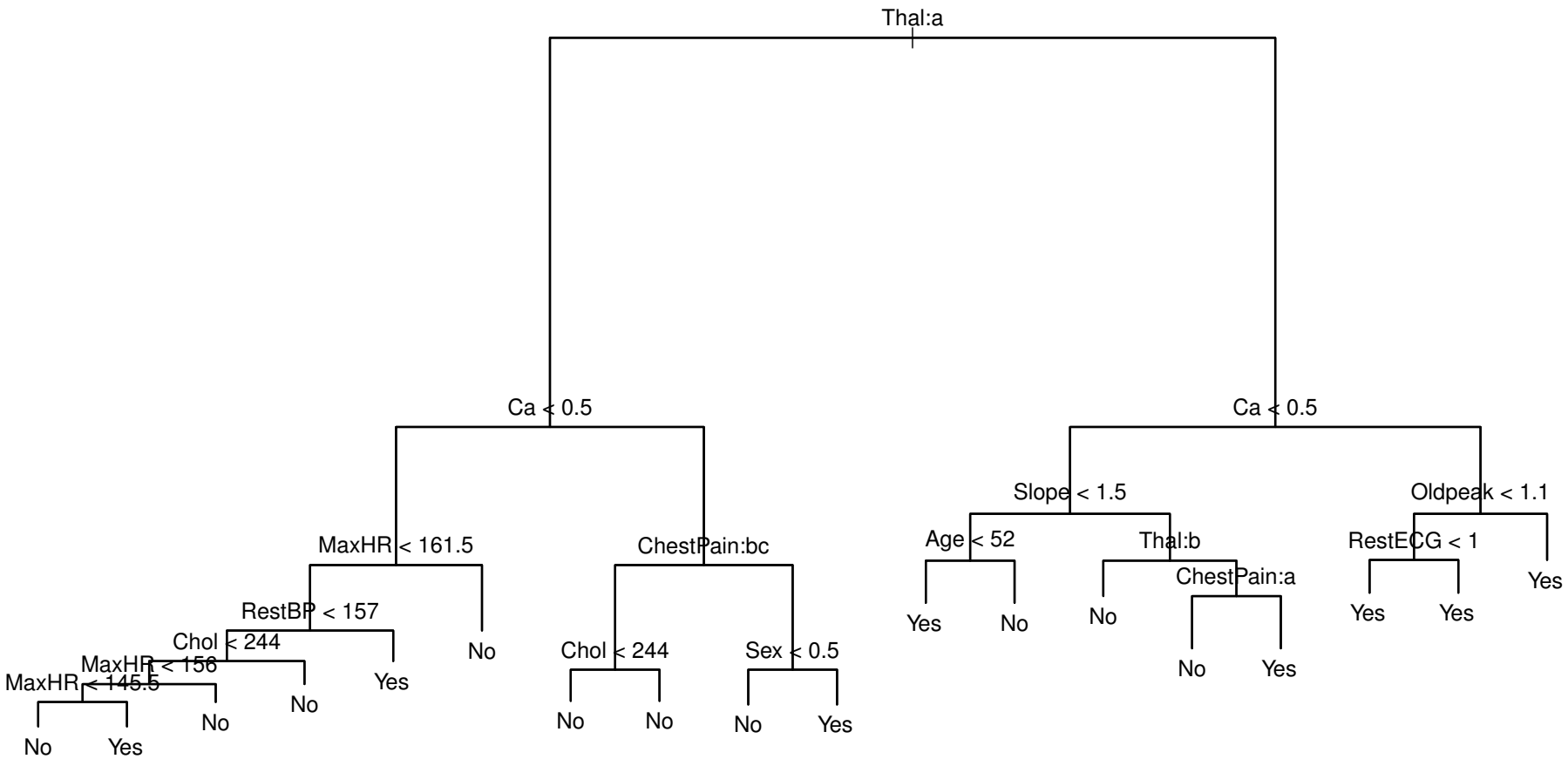
- Third choice: cross-entropy (similar to Gini index)

Classification Tree: Fitting

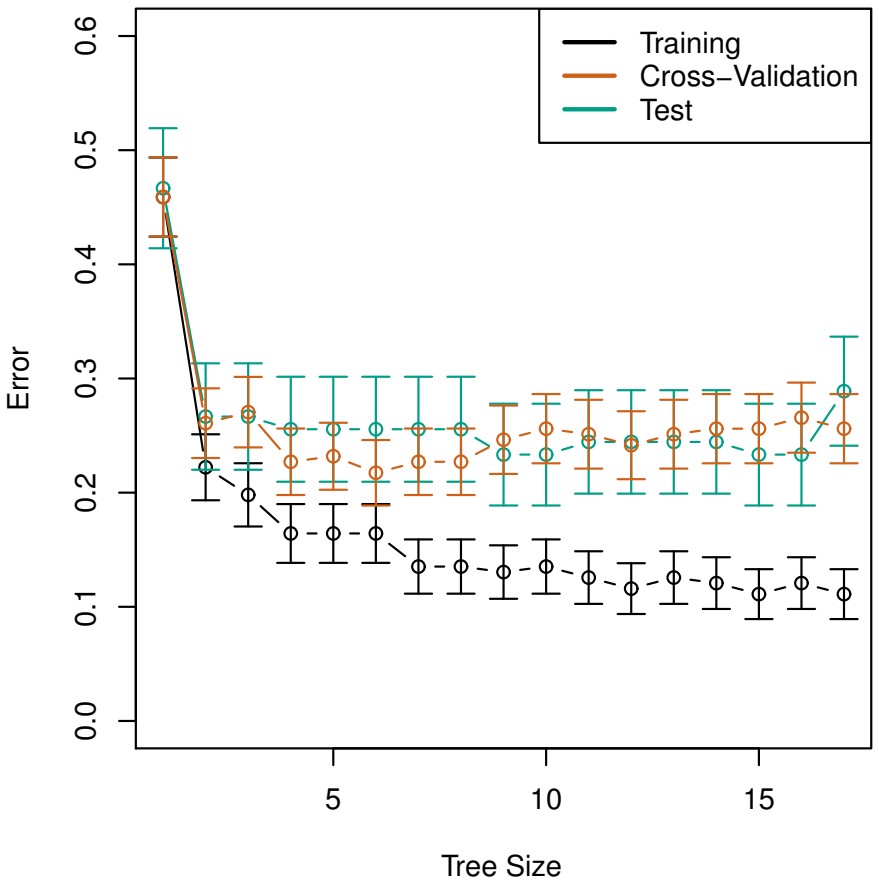
Algorithm:

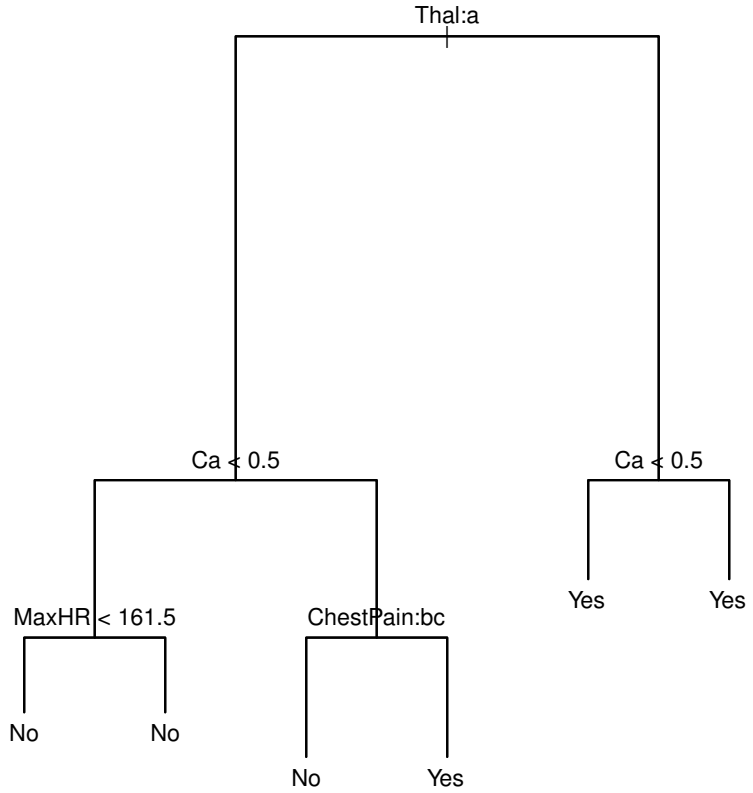
1. Grow a large tree by recursive binary splitting using Gini index.
   Stop when a leaf/region has too few observations
2. For each value of α, obtain a subtree by cost complexity pruning using classification error
3. Compute the CV error of the subtree corresponding to each α.
   Pick and output the best subtree

Heart data: Growing a classification tree



Heart data: Pruning a classification tree





Trees vs. Linear Models

Consider classification

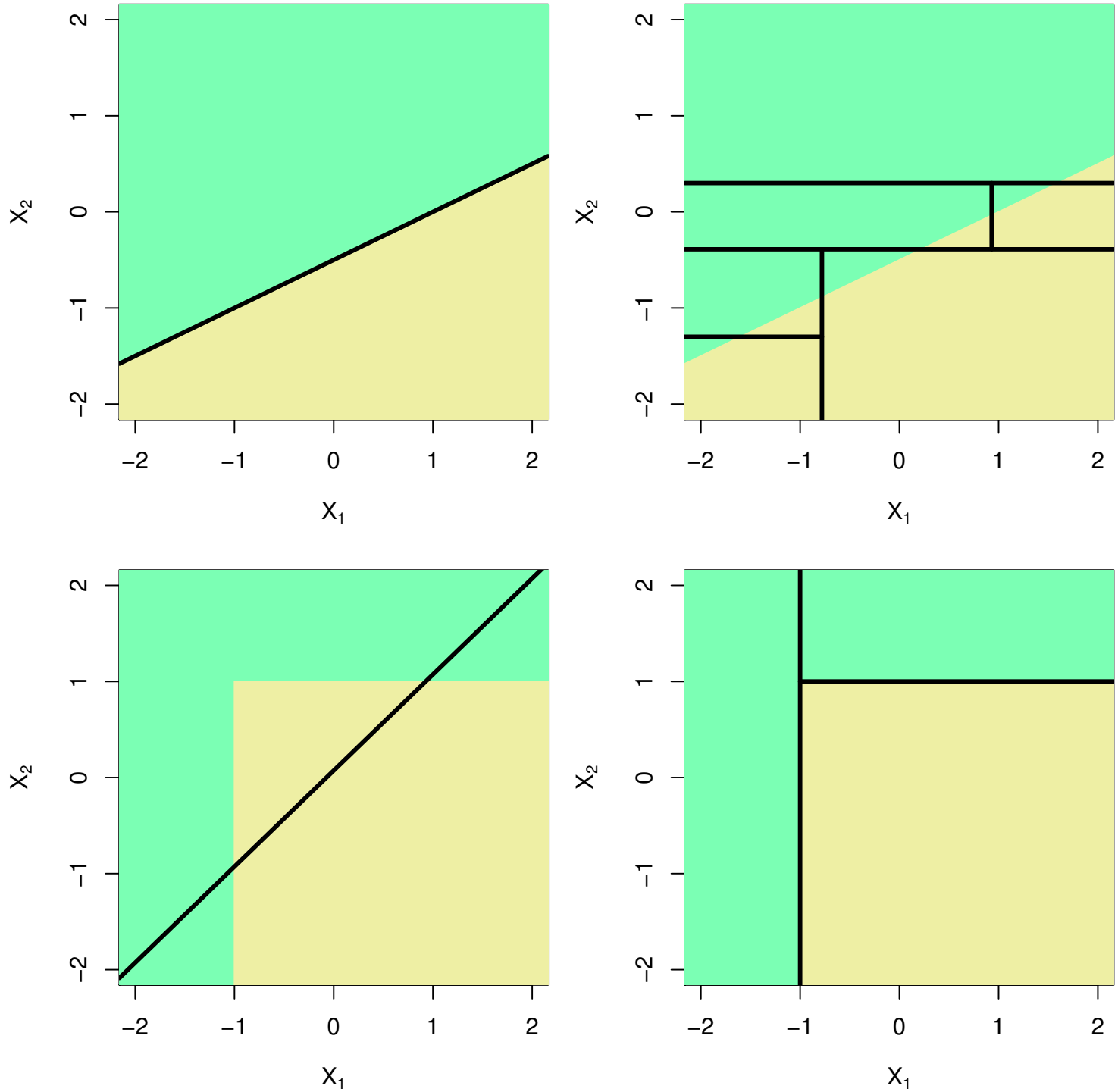
- A linear model (logistic regression) partitions the predictor space by a linear boundary

$$\log\left(\frac{\hat{\Pr}(Y=1)}{1-\hat{\Pr}(Y=0)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

- A decision tree partitions the predictor space into boxes

They work well under different scenarios

Trees are easy to interpret and visualize


X2
X1
−2
−1
0
1
2

Decision Trees: Pros and Cons

- Easy to visualize
- Easy to interpret and explain
- Easily handles categorical predictors
- Some believe that decision trees are similar to decision-making by human

- A single decision tree is often not very accurate

Bagging

A single tree often has high variance

To reduce variance: average many decision trees

- Ideally: Fit trees to many training sets
- Alternatively: Divide a training set into B (non-overlapping) subsets

Bagging: Bootstrap Aggregating ISLR 8.2.1

Bagging: Sample B overlapping sets with replacement

- Grow a large regression tree $\hat{f}^{*b}(\cdot)$ based on the b-th subset
- Then average

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Bagging: Bootstrap Aggregating

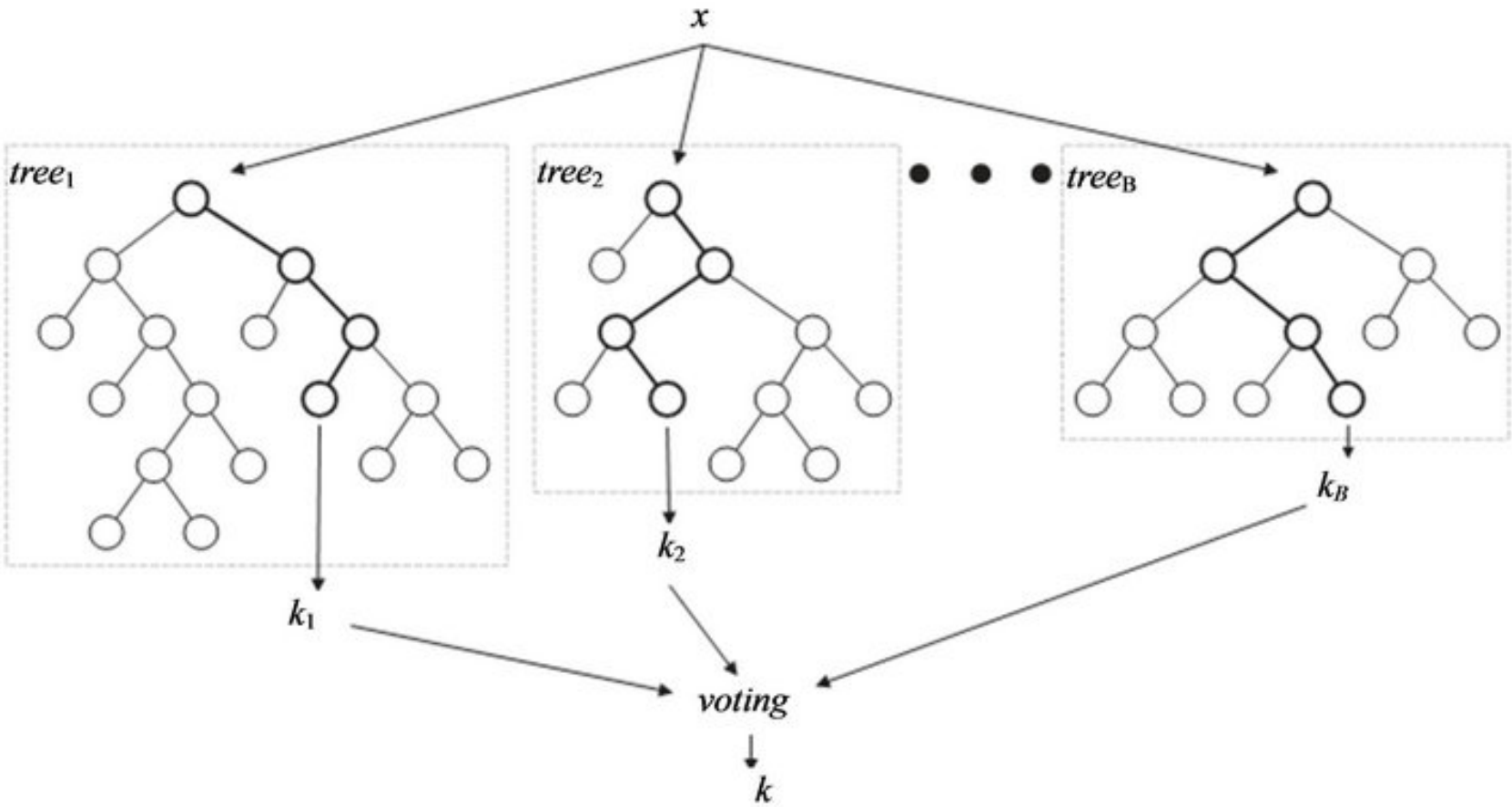
For regression:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

For classification:

$$\hat{f}_{\text{bag}}(x) = \text{majority-vote}\left(\hat{f}^{*1}(x), \hat{f}^{*2}(x), \ldots \hat{f}^{*B}(x)\right)$$

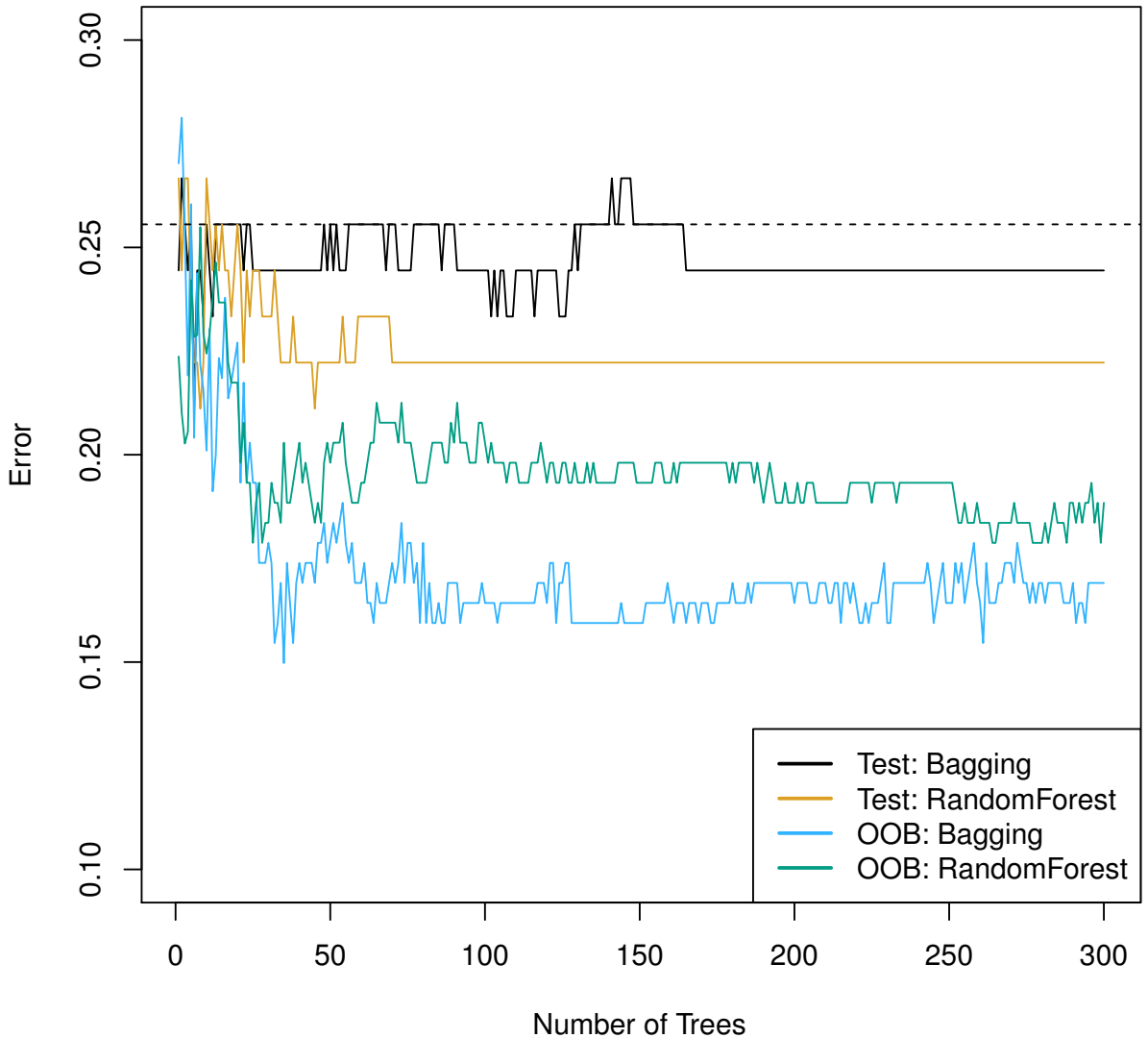
Bagging: Bootstrap Aggregating



(Credit: Cuong Nguyen, Yong Wang, Ha Nam Nguyen 2013)

Heart Dataset

Response: Heart disease or not
13 Predictors: Age, Sex, ChestPain, MaxHR, Thal, Ca . . .



Bagging Tree Algorithm

Given: training data (X, Y)

---

1. For $b = 1, \ldots, B$
   - **a** Sample a set (X^{*b}, Y^{*b}) with replacement from (X, Y)
   - **b** Grow a large tree f^{*b} based on (X^{*b}, Y^{*b})
2. Aggregate the B trees by

   (for regression)

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

   (for classification)

$$\hat{f}_{\text{bag}}(x) = \text{majority-vote}\left(\hat{f}^{*1}(x), \hat{f}^{*2}(x), \ldots \hat{f}^{*B}(x)\right)$$

- Each tree f^{*b} should be large (no pruning)
- B (# trees) not critical; use large values (hundreds or thousands)

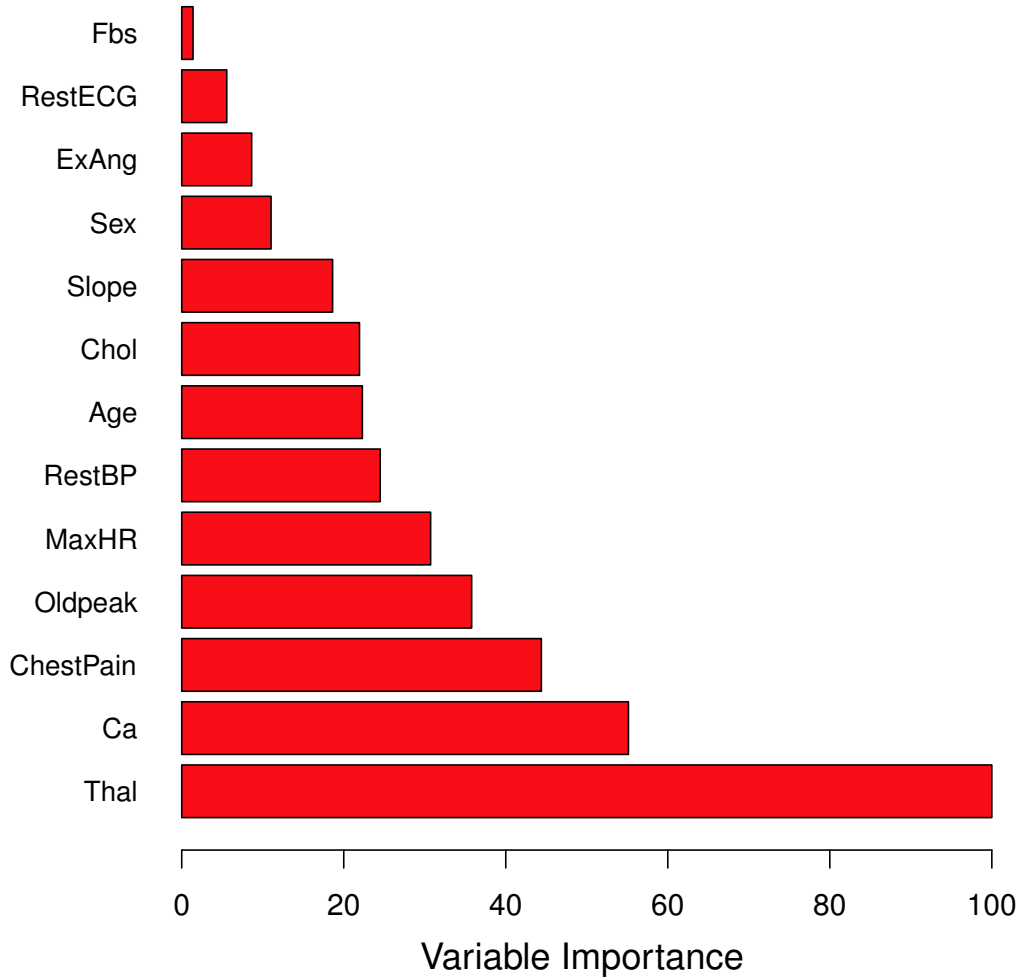
Interpreting Bagging Trees

Bagging

- Improve accuracy over a single tree
- Harder to interpret

Importance of variable *j*:

Reduction of RSS (or Gini index) due to splitting over variable *j*, average over *B* trees



Random Forests

Bagging:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Averaging reduces variance

Averaging uncorrelated quantities reduces variance!

But the trees $\hat{f}^{*1}, \hat{f}^{*2}, \ldots, \hat{f}^{*B}$ may be very correlated.
Example: A strong predictor/feature

Random Forest: decorrelates trees by random feature selection

Random Forests

Growing a tree $\hat{f}^{*b}$:

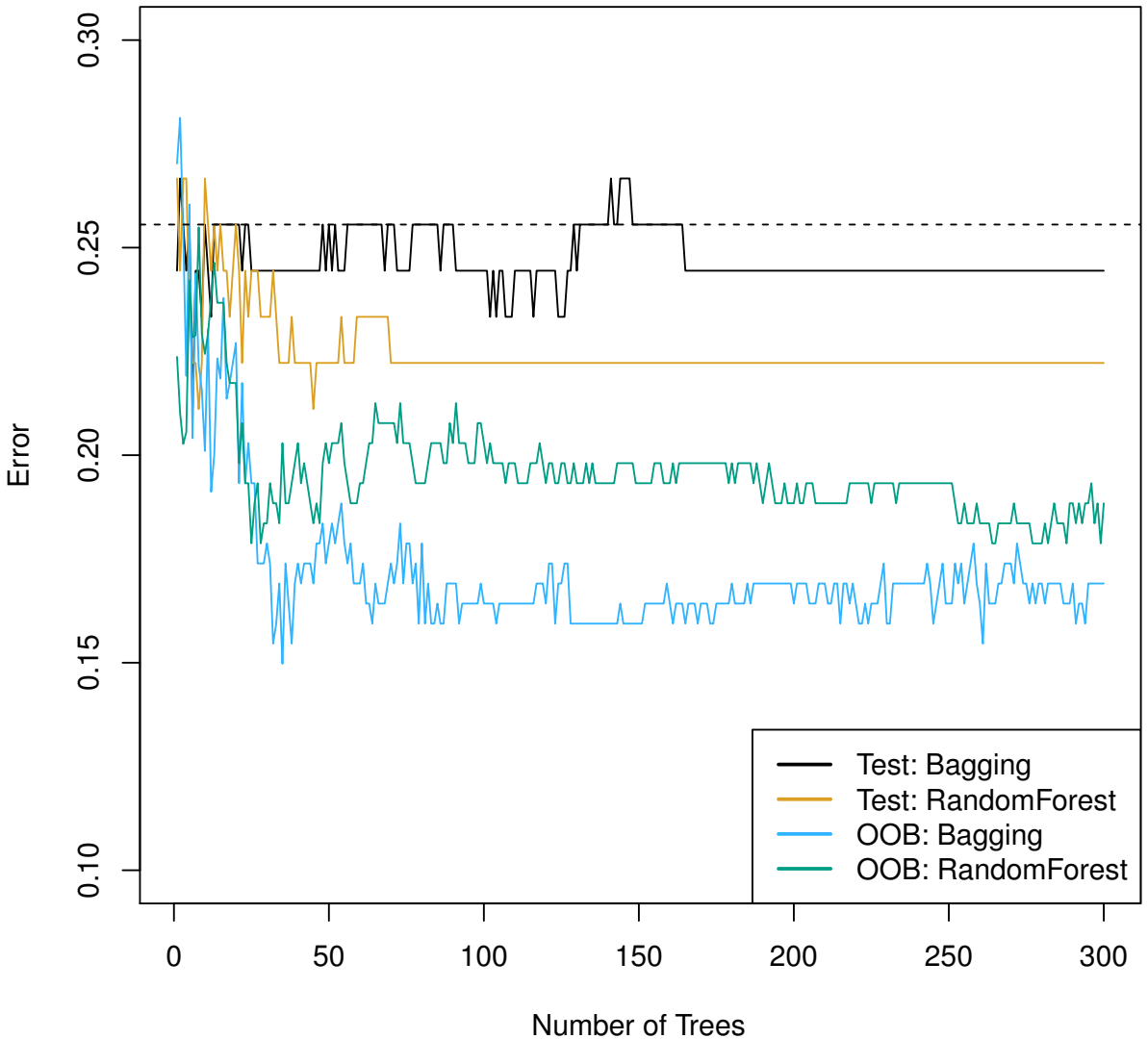
Each time a split is to be identified

- Out of p predictors, randomly sample $m \ll p$ of them as candidates
- Find the best split based on one of these m predictors

- Trees are decorrelated $\Rightarrow$ Reduced variance
- Popular: $m = \sqrt{p}$
- If $m = p$: same as bagging

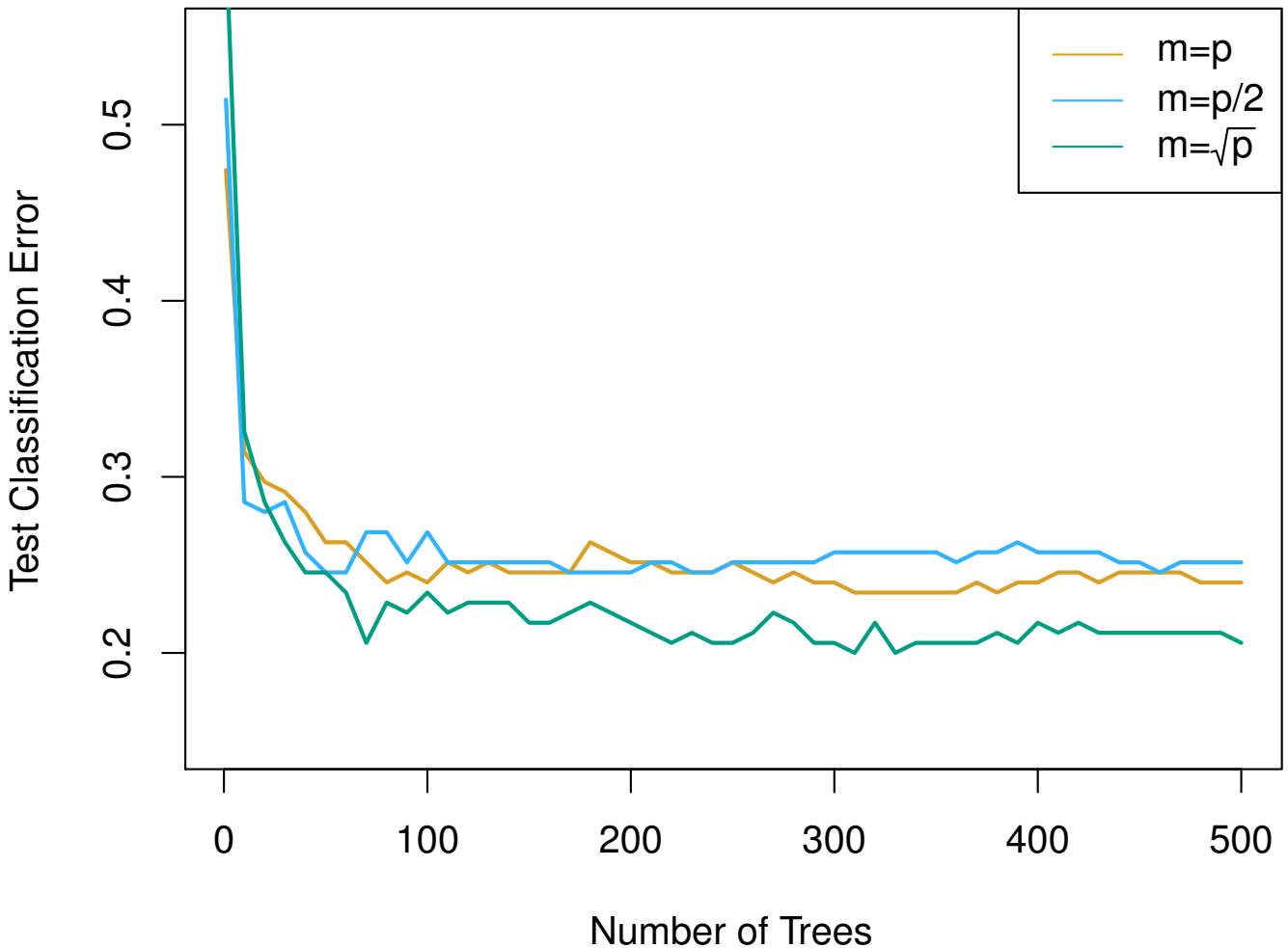
Example 1: Heart Dataset

Response: Heart disease or not
13 Predictors: Age, Sex, ChestPain, MaxHR, Thal, Ca . . .



Example 2: Cancer Dataset

- $n = 349$ observations (patients)
- $p = 500$ predictors (gene measurements)
- Response: Normal, Cancer 1, Cancer 2, . . . , Cancer 14



Boosting and Bagging:

- Combine many trees
- Can be applied to improve other learning methods

Boosting: combine simple trees sequentially

Boosting (Optional) ISLR 8.2.3

Idea: Fit a new tree to the residual of the previous model

Algorithm (for boosting regression trees)

1. Initialization: $\hat{f}(x) \equiv 0$ and $r_i = y_i$, $i = 1, 2, \ldots, n$
2. For $b = 1, 2, \ldots, B$:
   - **a** Fit residual: Fit a new tree $\hat{f}^b$ with $d + 1$ leaves to the data X, r
   - **b** Shrink and combine: Update $\hat{f}$ by adding a shrunken version of $\hat{f}^b$:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

   - **c** Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i), \quad i = 1, 2, \ldots, n$$

3. Output $\hat{f}$

Key parameters: #trees B, tree complexity/depth d, shrinkage rate λ

Boosting

2 For $b = 1, 2, \ldots, B$:

a Fit residual: Fit a new tree $\hat{f}^b$ with $d + 1$ leaves to the data X, r

b Shrink and combine: Update $\hat{f}$ by adding a shrunken version of $\hat{f}^b$:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

c Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i), \quad i = 1, 2, \ldots, n$$

Philosophy: Learn slowly

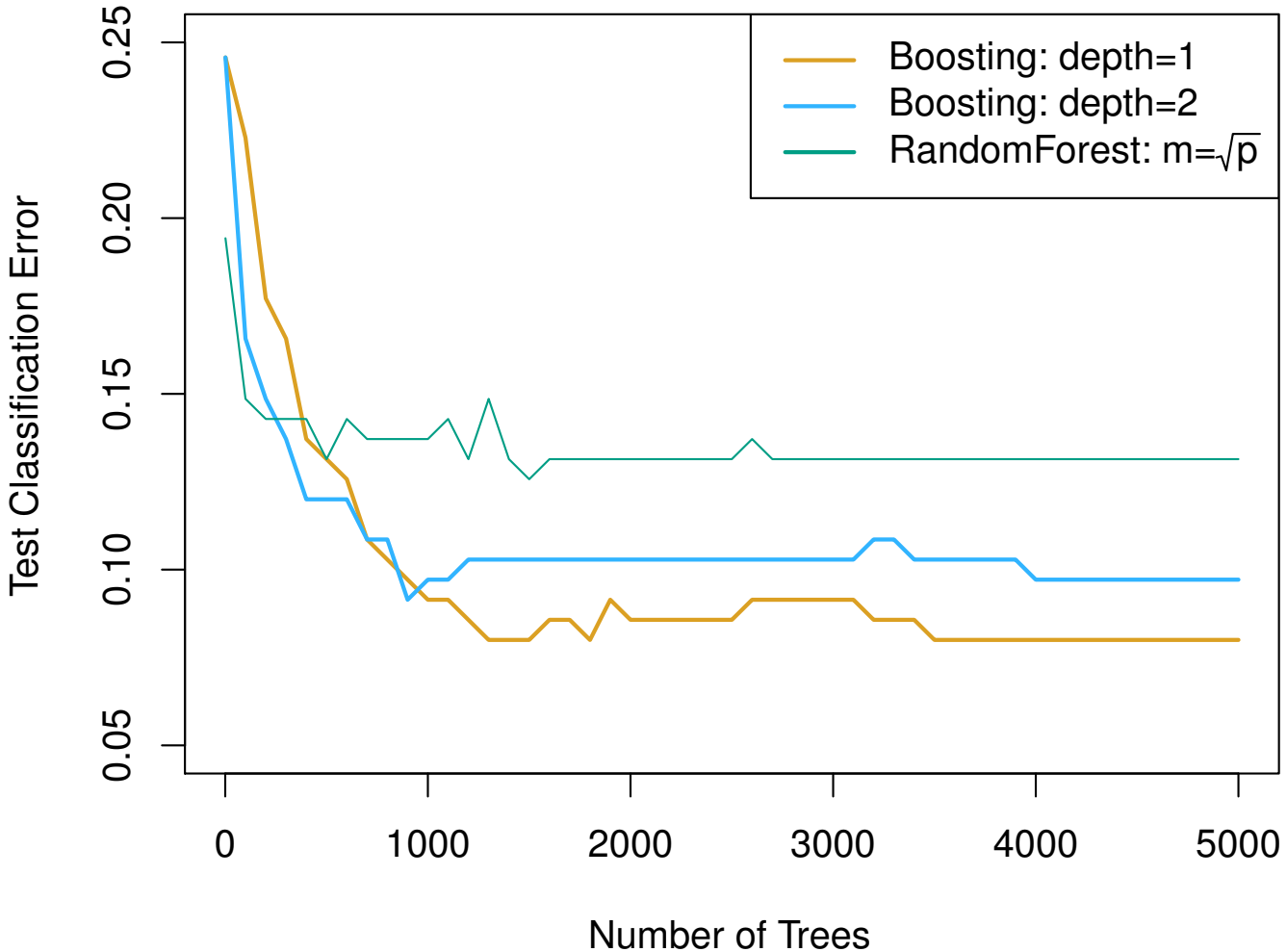
- Each time fit a simple tree (with small d) to the residual
- Combine slowly with a small λ

Choosing parameters:

- #trees B: by CV (large B may overfit)
- Shrinkage rate λ: 0.01 or 0.001
- Depth/complexity d of each tree: $d = 1, 2, 3, 4$

Example: Cancer dataset

- $n = 349$ observations (patients)
- $p = 500$ predictors (gene measurements)
- Response: Normal, Cancer 1, Cancer 2, . . . , Cancer 14



Random Forests vs. Boosting (Optional)

- Combine many trees into one model
- Improve over a single decision tree

- RF: use large trees (grow without pruning)
- Boosting: use small trees (small d)

- RF: train trees on bootstrap samples
- Boosting: train trees on residuals

- RF: combine by averaging/voting
- Boosting: combine by adding trees sequentially

- RF: take B large; won't overfit
- Boosting: choose B by CV; large B may overfit

Random Forests vs. Boosting (Optional)

Two "opposite" ways of improving a single tree

RF: easy to tune; usually don't overfit

Boosting: hard to tune; may overfit
but well-tuned boosting trees often outperform RF