# Tree Based Methods

## Section 8.1

Cathy Poliak, Ph.D.
cpoliak@central.uh.edu

Department of Mathematics
University of Houston

# Tree Based Models

So far we have covered such (relatively) basic models as:

- Linear Regression
- Logistic Regression - response - categorical, classification

and such resampling techniques as

- Cross-Validation
- Bootstrap

we are ready for another model class:

- **Tree-based models**.

Tree-based models can be applied to **both** regression and classification problems.

# Motivating example

**Example.** For *Hitters* data on baseball hitters, we'd like to predict baseball player's *Salary* based on

- *Years* - # years he played in major leagues, and
- *Hits* - # hits made in the previous year.

```
> head(Hitters[,c("Salary","Years","Hits")])
Salary Years Hits
-Andy Allanson         NA     1   66
-Alan Ashby         475.0    14   81
-Alvin Davis        480.0     3  130
-Andre Dawson       500.0    11  141
-Andres Galarraga    91.5     2   87
-Alfredo Griffin    750.0    11  169
```

1. From our models that we have previous talked about, which would be best to use in this example?

a) Linear Regression b) Logistic Regression

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

$y_i$ = observed response value

$\hat{y}_i$ = predicted response value

```
> (mse.hitter.train = mean((log(Hitters2[train,]$Salary) - pred.train)^2))
[1] 0.3262283
> sqrt(exp(.3262))
[1] 1.177154
```

with the training set we are off by $1,177.15.

```
> pred.test = predict(lm.hitters,newdata = Hitters2[-train,])
> (mse.hitter.test = mean((log(Hitters2[-train,]$Salary) - pred.test)^2))
[1] 0.4696672
> sqrt(exp(0.4697))
[1] 1.264719
```

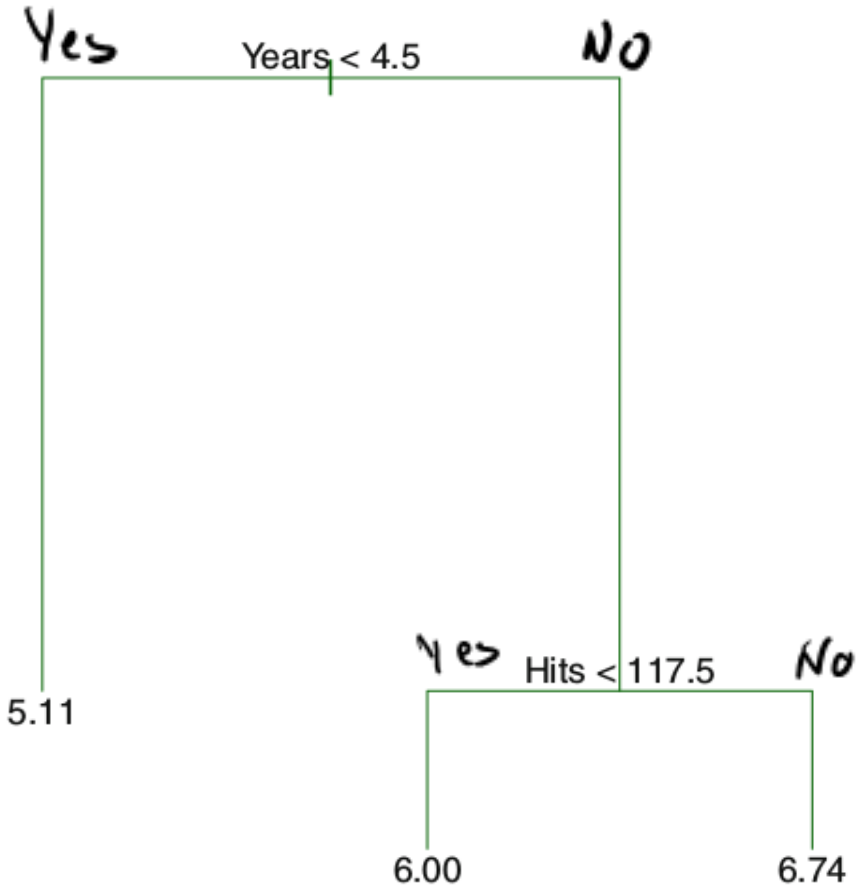
For the test data we are off by about $1,264.72.

# Use A Decision Tree Instead

After certain data pre-processing steps, such as:

- dropping observations with missing values
- applying log-transformation to response variable *Salary*

an example of a fitted decision tree looks as follows



# Regression Trees: *Hitters* example

Explanation of the tree.

– Predictor space (range of values for *Years* and *Hits* variables) got segmented into **3** regions (terminal nodes):

- $R_1 = (Years < 4.5)$
- $R_2 = (Years \geq 4.5) \ \ \& \ \ (Hits < 117.5)$,
- $R_3 = (Years \geq 4.5) \ \ \& \ \ (Hits \geq 117.5)$

# Regression Trees: *Hitters* example

Explanation of the tree.

– Predictor space (range of values for *Years* and *Hits* variables) got segmented into **3** regions (terminal nodes):

- $R_1 = (Years < 4.5)$
- $R_2 = (Years \geq 4.5)$ & $(Hits < 117.5)$,
- $R_3 = (Years \geq 4.5)$ & $(Hits \geq 117.5)$

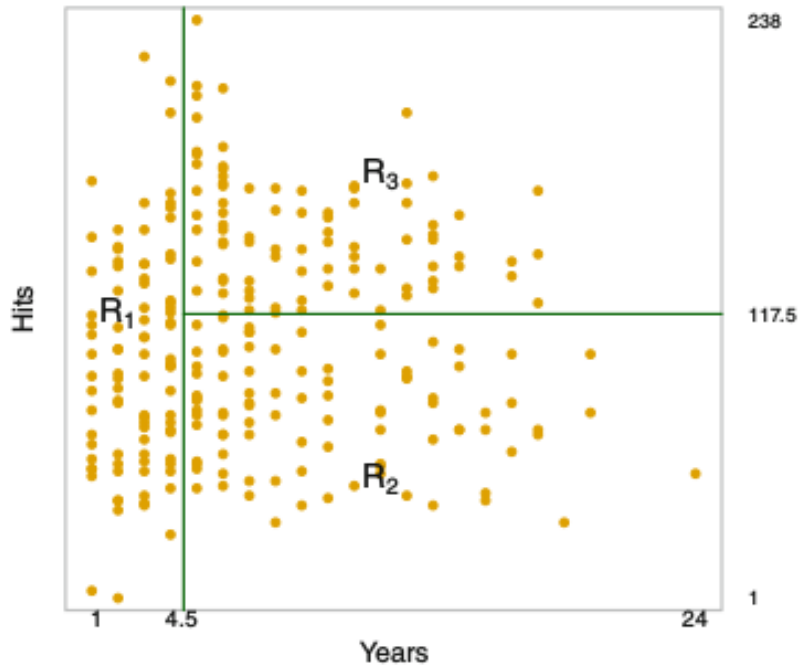
– *Salary* prediction in region $R_i$ = (**mean** *Salary* of **all** hitters $\in$ $\boldsymbol{R_i}$):

- Any hitter in $R_1$ ($< 4.5$ years of experience) is predicted to make $e^{5.11} \approx 165k$.
- For hitters with over 4.5 years of experience:
  - if hitter $\in R_2$ ($< 117.5$ hits), he is projected to make $e^{6.00} \approx 403k$,
  - while for $R_3$ ($\geq 117.5$ hits) - $e^{6.74} \approx 845k$

# Regression Trees: General Idea

Decision trees for regression is also known as **regression trees**, are built via:

- Segmenting predictor space ($\leftrightarrow$ set of all possible values for $X_1, \ldots, X_p$) into simple non-overlapping regions $R_1, \ldots, R_J$.

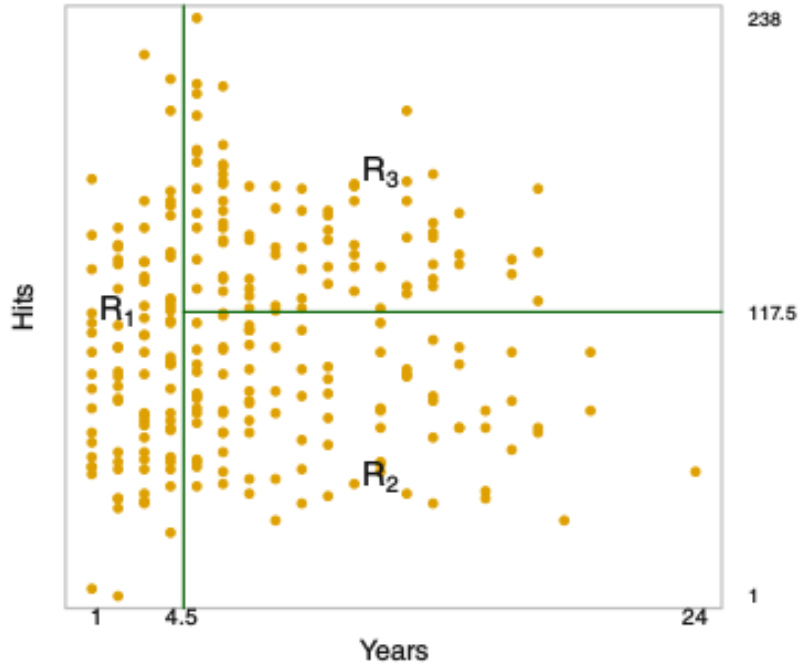



- For every observation falling into region $R_j$, the predicted response is the **mean response** of *all* training observations in $R_j$.

# Back to *Hitters*: How do we segment?

How do we construct the regions $R_1, \ldots, R_J$ ?

1. Regions are built via splitting range of a variable. E.g. range of *Years* is split **at value** 4.5 into $(Years < 4.5)$ and $(Years \geq 4.5)$.




Results are boxes, or high-dimensional rectangles, for the sake of

- simplicity,
- ease of interpretation

# How do we segment? Recursive binary splitting.

2. We need a partition $R_1, \ldots, R_J$ that minimizes

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

It is computationally infeasible to go through all possible partitions $R_1, \ldots, R_J$

# How do we segment? Recursive binary splitting.

2. We need a partition $R_1, \ldots, R_J$ that minimizes

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

It is computationally infeasible to go through all possible partitions $R_1, \ldots, R_J$

**Recursive binary splitting** is an approach that's

- top-down (begins with a full region and then successively splits the predictor space)
- greedy (at each step the best split is made regardless of next steps)

# Recursive Binary Splitting: Steps

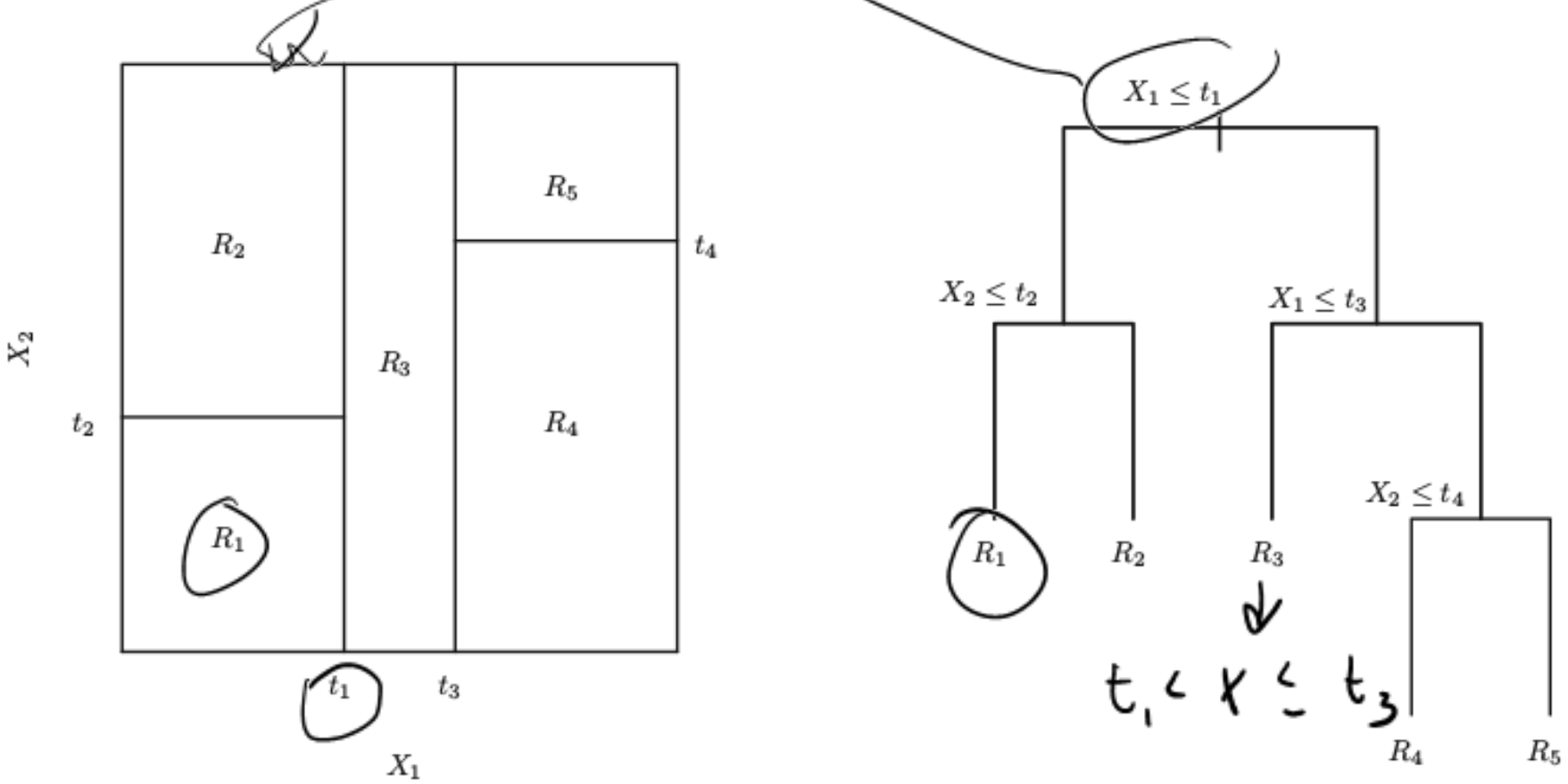
**Notation**: $R(j, s) = \{X \mid X_j < s\}$ is the region of predictor space $(X_1, \ldots, X_p)$ where $X_j < s$ ($j^{th}$ predictor is less than value $s$).

**Steps** of recursive binary splitting:

1. Start with full predictor space $R = \{(X_1, \ldots, X_p)\}$.
2. For any $j$ and $s$, we define the pair of half-planes
$$R_1(j, s) = \{X | X_j < s\}, \quad R_2(j, s) = \{X | X_j \geq s\},$$
3. We seek for values of $j$ and $s$ that minimize
$$RSS = \sum_{i:\ x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$
4. We repeat steps 2 and 3 trying to split the data further by minimizing the RSS, until a stopping criterion (e.g. "no region contains more than five observations") is reached .
5. We get a final set of regions $R_1, \ldots, R_J$, and later predict the **response** for a test observation from region $R_j$ via the **mean** of training observations $\in R_j$, $j = 1, \ldots, J$.

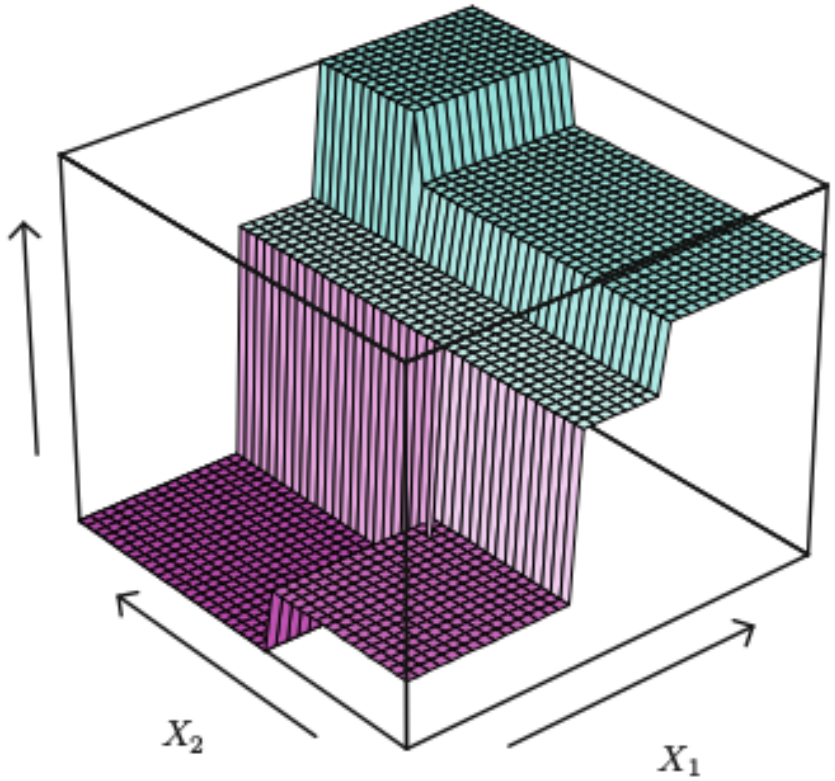
# Recursive Binary Splitting: Simulated Example

**Example.** Below is the output of recursive binary splitting applied to a simulated two-dimensional example with full predictor space $\{(X_1, X_2)\}$. It resulted into five regions, which can be seen on the left, and the corresponding tree is on the right.



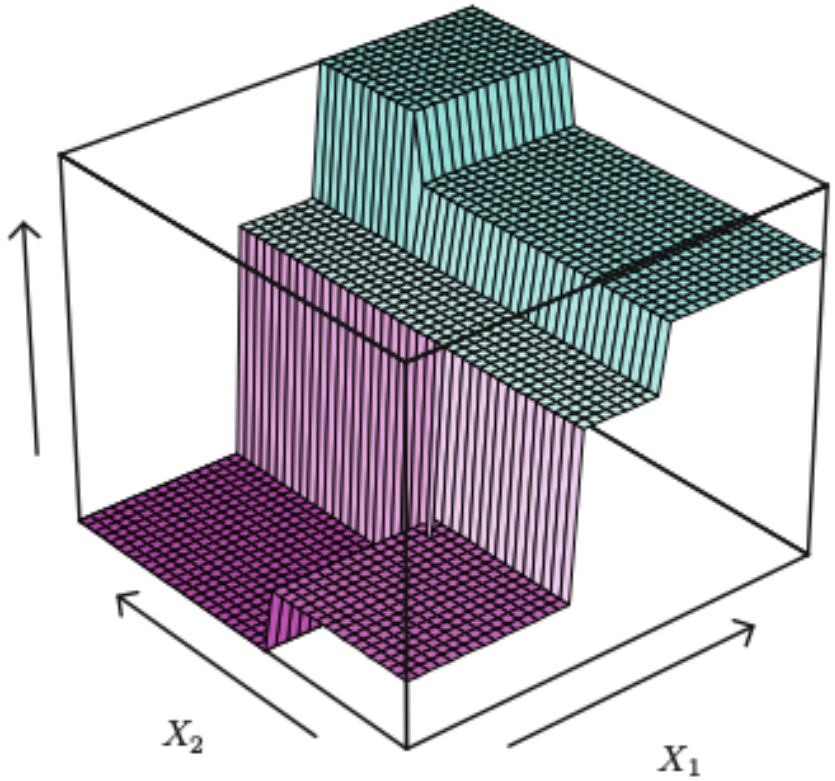
# Recursive Binary Splitting: Simulated Example

Prediction **surface** corresponding to that tree is depicted below.



# Recursive Binary Splitting: Simulated Example

Prediction **surface** corresponding to that tree is depicted below.




*z*-axis contains predicted response value for the respective region $R_j$.

# Overfitting Issue: Tree Pruning

Recursive binary splitting usually results into large trees that

- produce good predictions on training data, but
- tends to overfit and perform poorly on test data.

A smaller tree with fewer splits ($\leftrightarrow$ fewer regions $R_1, \ldots, R_J$) might

- lead to lower variance and better testing set performance,
- better interpretation,
- at the cost of a little extra bias.

# Overfitting Issue: Tree Pruning

Recursive binary splitting usually results into large trees that

- produce good predictions on training data, but
- tends to overfit and perform poorly on test data.

A smaller tree with fewer splits ($\leftrightarrow$ fewer regions $R_1, \ldots, R_J$) might

- lead to lower variance and better testing set performance,
- better interpretation,
- at the cost of a little extra bias.

**Solution**:

1. Grow a large tree $T_0$, but then
2. **Prune** it back in order to obtain a subtree.

For step 2 our goal is to select a subtree that leads to the lowest **test** error rate (which could be estimated via cross-validation).

This is infeasible (too many subtrees) for every possible subtree.

# Overfitting Issue: Cost Complexity Pruning

In **cost complexity** pruning, instead of all possible subtrees, we consider a sequence of trees indexed by a tuning parameter $\alpha \geq 0$.

For each value of $\alpha$ there's a subtree $T \subset T_0$ that minimizes

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

where $|T|$ - number of terminal nodes of tree $T$; $R_m$ - region for $m^{th}$ terminal node; $\hat{y}_{R_m}$ - predicted response for $R_m$ (mean of obs. $\in R_m$).

# Overfitting Issue: Cost Complexity Pruning

In **cost complexity** pruning, instead of all possible subtrees, we consider a sequence of trees indexed by a tuning parameter $\alpha \geq 0$.

For each value of $\alpha$ there's a subtree $T \subset T_0$ that minimizes

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

where $|T|$ - number of terminal nodes of tree $T$; $R_m$ - region for $m^{th}$ terminal node; $\hat{y}_{R_m}$ - predicted response for $R_m$ (mean of obs. $\in R_m$).

Tuning parameter $\alpha$ controls the trade-off between

- quality of fit the subtree $T$ provides to training data (left part), and
- the complexity ($\leftrightarrow$ # of terminal nodes) of subtree $T$ .

As $\alpha$ increases, there's a penalty for trees with too many terminal nodes $\implies$ quantity (1) tends to be minimized for a smaller subtree.

# Decision Tree: Full Algorithm

Below are the steps of building a decision tree (ISLR, page 309):

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of $\alpha$.

$|T|$ = # of terminal nodes

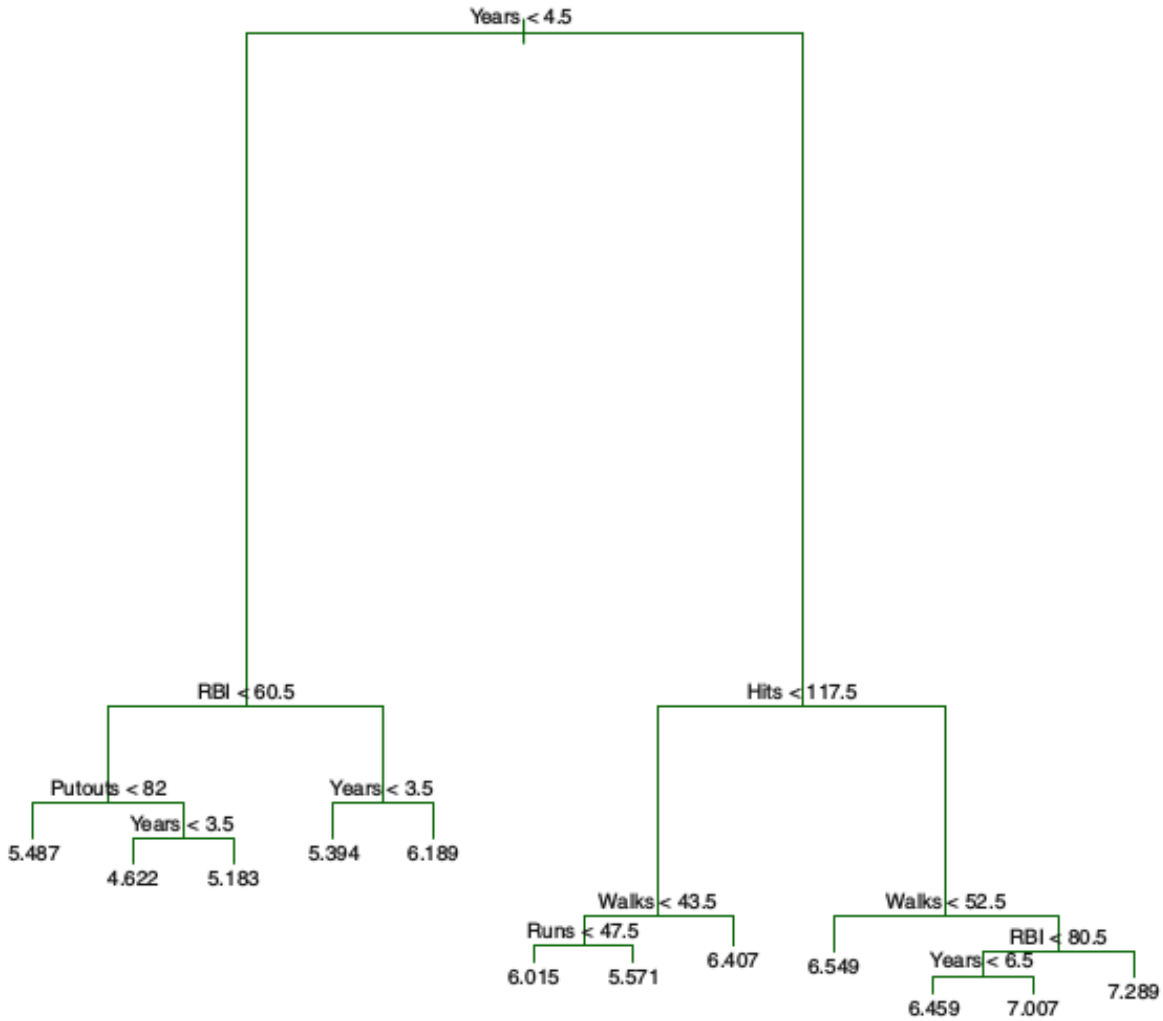
3. Use $K$-fold cross validation to choose $\alpha$. That is, divide the training observations into $K$ folds. For each $k = 1, \ldots, K$;
   (a) Repeat steps 1 and 2 on all but the $k$th fold of the training data.
   (b) Evaluate the mean squared prediction error on the data in the left-out $k$th fold, as a function of $\alpha$.

4. Average the results for each value of $\alpha$, and pick $\alpha$ to minimize the average error.

5. Return the subtree from Step 2 that corresponds to the chosen value of $\alpha$.

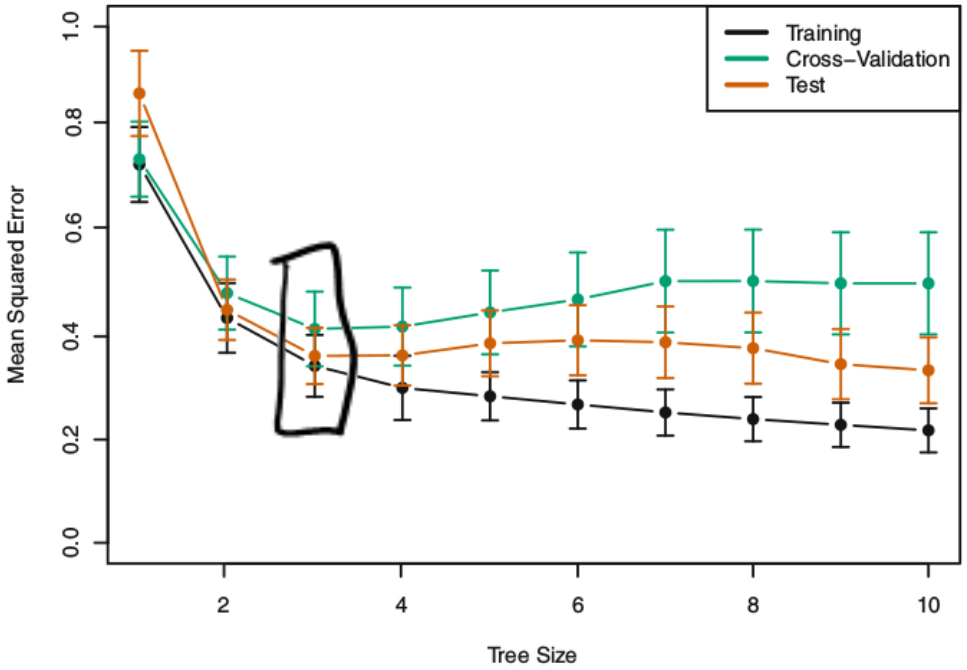
# *Hitters* example: Large tree

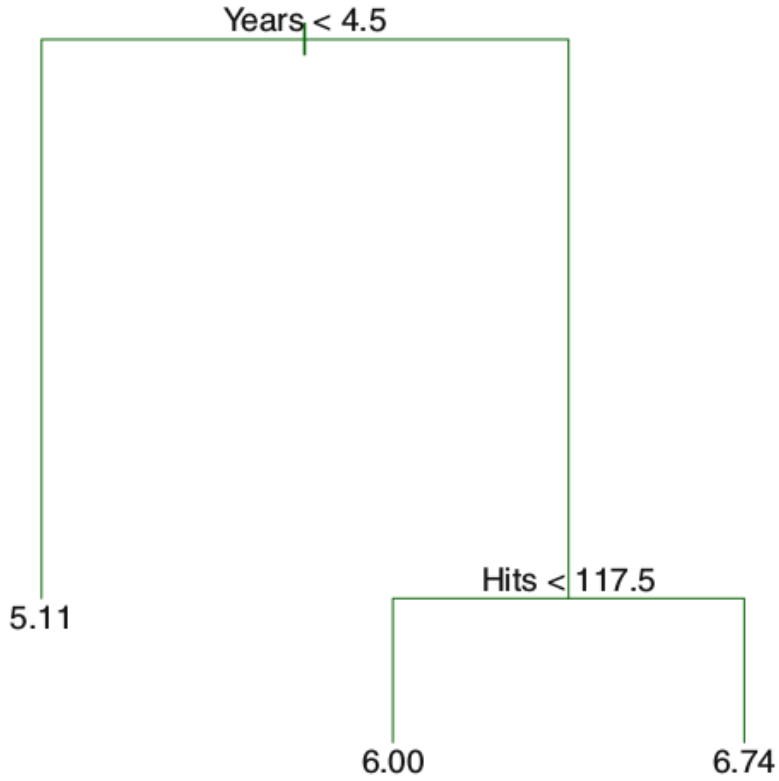
**Example.** We randomly divide *Hitters* data into 132 training and 131 testing observations, and build a large regression tree on 9 features:



# *Hitters* example: Pruning

Then we perform pruning, and record **training error**, **CV error** & actual **test set error** for all resulting subtree sizes.




CV yields a subtree with $|T| = 3$ terminal nodes as the optimal one.

# R Code for Decision Trees

- Uses the tree package
- The following is to set up the data.

```
install.packages('tree')
library(tree)
library(ISLR)
Hitters2 = na.omit(Hitters)
```

# `tree` Function

Type in a run the following

```
set.seed(100)
train = sample(1:nrow(Hitters2),nrow(Hitters2)/2)
tree.hitters = tree(log(Salary) ~ Hits + HmRun + Runs + RBI + Walks +
                                  Years + PutOuts + Assists + Errors
                    Hitters2,subset = train)
summary(tree.hitters)
```

2. How many "nodes" did this produce?

   a) 10 b) 9 c) 4 d) 3

3. Type and Run the following in R

   ```
   plot(tree.hitters)
   text(tree.hitters)
   ```

   What is the predicted salary for a player that has between 3.5 and 4.5 *Years* and *RBI* greater than 43.5?

   a) \$5.885
   b) \$5,885
   c) \$359.60
   d) \$359,602.80 = exp(5.885)

# Pruning the Tree

We think that 11 regions are too many. We can do the following to see how many regions we need.

```
cv.hitters = cv.tree(tree.hitters)
plot(cv.hitters$size,cv.hitters$dev,type = "b")
```

Where there is an apparent "elbow" determines how many regions we need.

4. How many regions (nodes) do we need?

   a) 11 b) 9 c) 4 d) 3

5. Type and Run the following in R

```
prune.hitters = prune.tree(tree.hitters,best = 3)
plot(prune.hitters)
text(prune.hitters)
```

   What is the predicted salary for a player that has played for less than 4.5 *Years*?

   a) \$5.135
   b) \$5,135
   c) \$169.86
   d) \$169,864.30 $\approx \exp(5.135)$

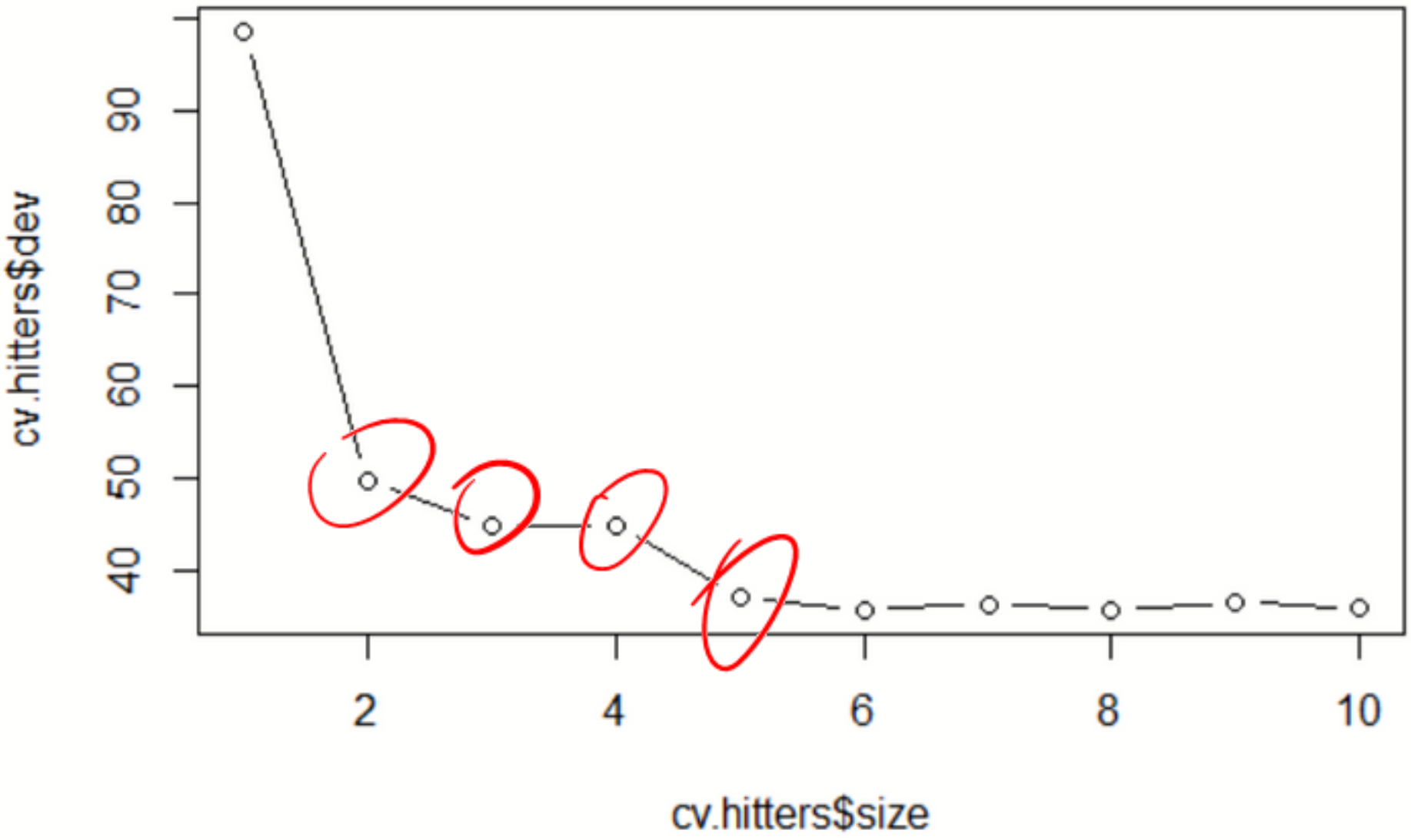
```
> cv.hitters = cv.tree(tree.hitters)
> cv.hitters
$size  # of terminal nodes
 [1] 10  9  8  7  6  5  4  3  2  1

$dev  MSE
 [1] 35.91129 36.45607 35.64970 36.12924 35.69940 37.02872 44.89761
44.68819
 [9] 49.70015 98.56832

$k
 [1]     -Inf  1.130695  1.245748  1.402796  2.256818  2.414474  4.540760
 [8]  5.188033  8.429781 47.849506

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```


cv.hitters$dev
40
50
60
70
80
90
2
4
6
8
10
cv.hitters$size

# How Well Are We Predicting

- We can use the test set to determine the MSE of both trees.
- For the original tree with 11 regions we get.

```
yhat = predict(tree.hitters,newdata=Hitters2[-train,])
hitters.test = Hitters2[-train,"Salary"]
plot(yhat,log(hitters.test))
abline(0,1)
mean((yhat-log(hitters.test))^2)
    [1] 0.4180251
```

- For the pruned tree we get

```
yhat.prune = predict(prune.hitters,newdata = Hitters2[-train,])
    plot(yhat.prune,log(hitters.test))
    abline(0,1)
    mean((yhat.prune - log(hitters.test))^2)
    [1] 0.4242652
```

- Taking the square root and exponential, we get
  - For the larger tree this model leads to test predictions that are within around $1,232 of the true salary for a player.
  - For the pruned tree this model leads to test predictions that are within around $1,236 of the true salary for a player.