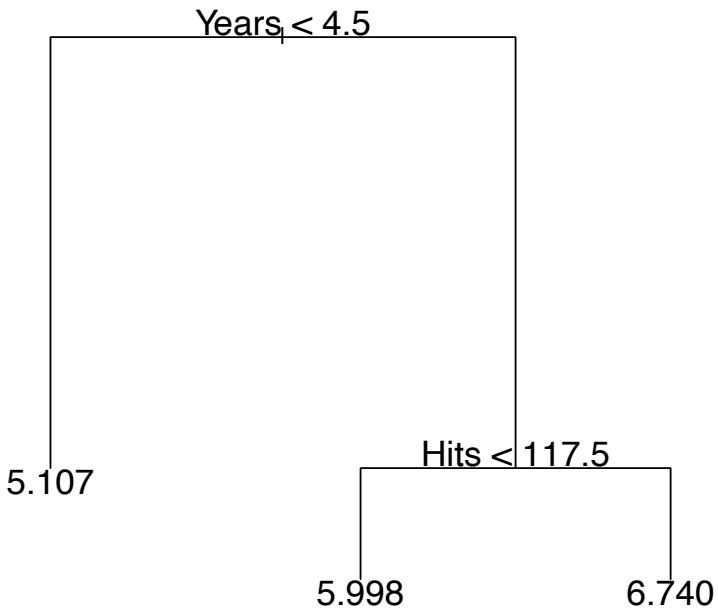
Tree-Based Methods

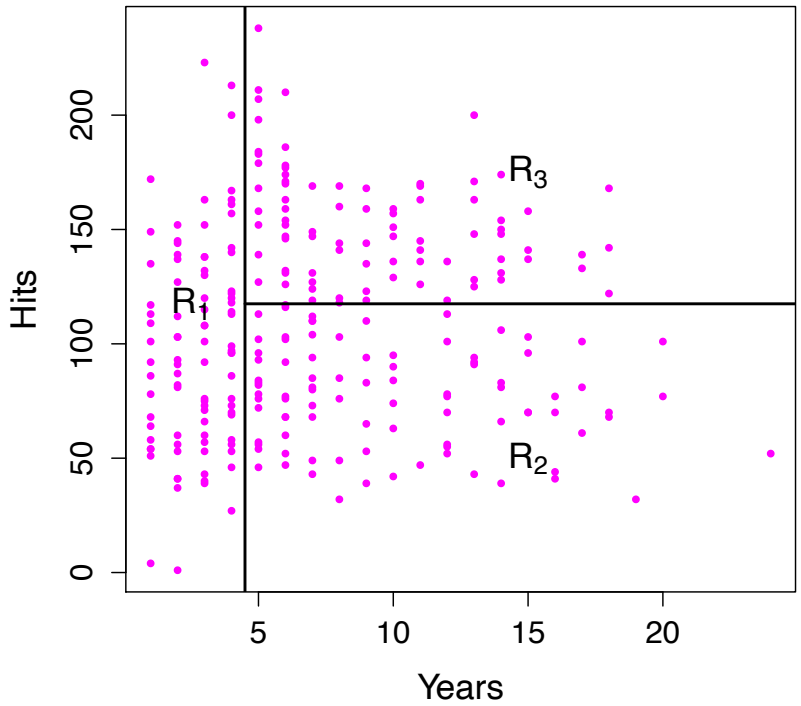
- Stratifying / segmenting the predictor space into a number of simple regions
- These types of approaches are known as *decision tree* methods
- Quantitative responses: **Regression trees**
- Qualitative responses: **Classification trees**

An Example – `Hitters` data

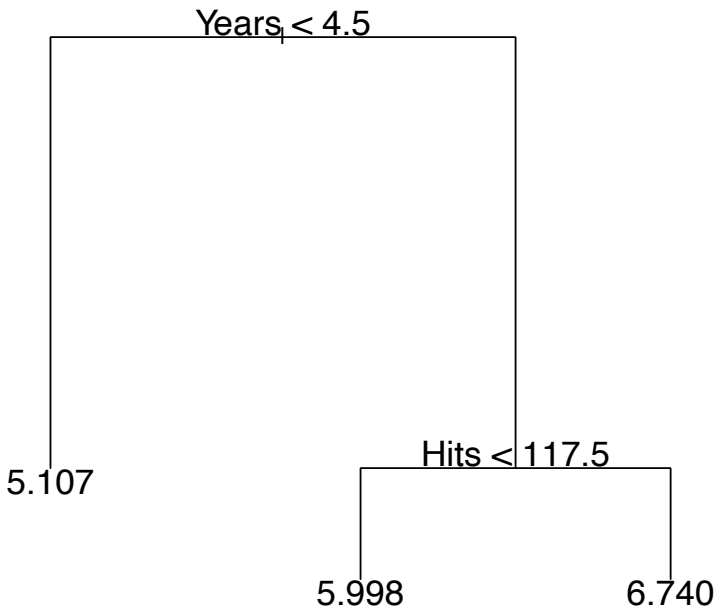
- Predict a baseball player's salary based on the number of years he has played in the major leagues and the number of hits he made in the previous year
- Here we log-transform the response variable

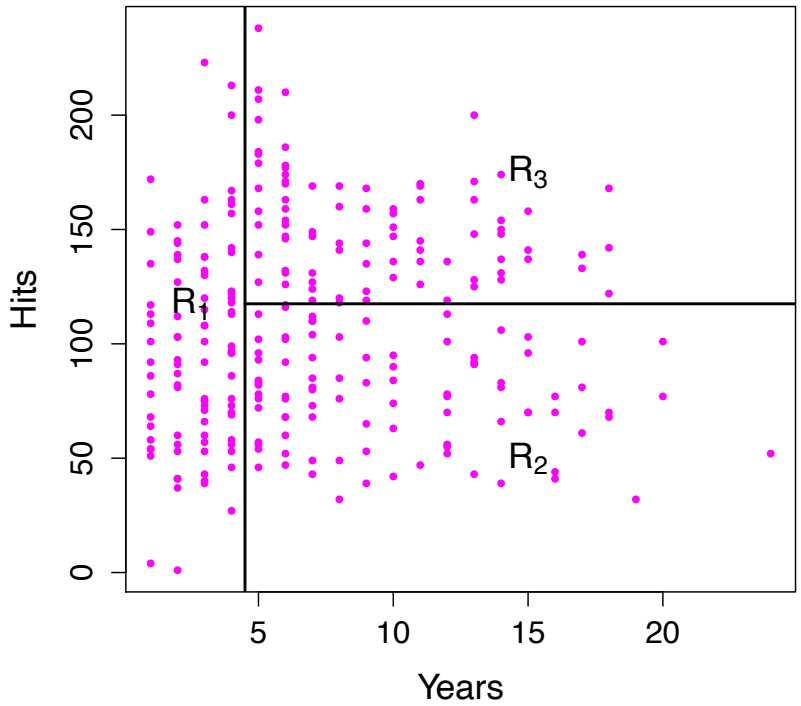
An Example – `Hitters` data





An Example – `Hitters` data





$$\$1,000 \times e^{5.107} = \$165,174$$

$$\$1,000 \times e^{5.998} = \$402,623$$

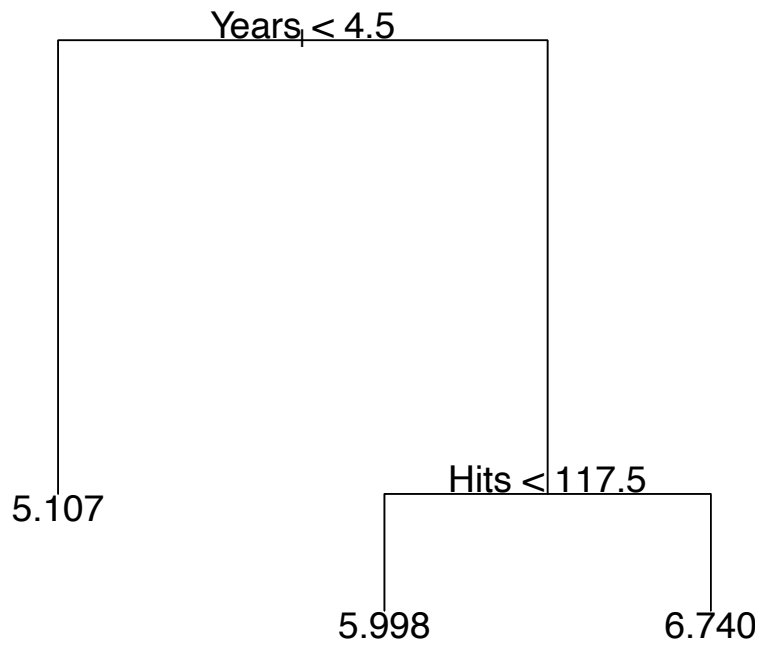
$$\$1,000 \times e^{6.740} = \$845,561$$

An Example – `Hitters` data

- Regions R_j are known as *terminal nodes* or *leaves* of the tree
- The points at which the predictors are split are referred to as *internal nodes*
- Segments that connect the nodes are called *branches*
- How do we interpret the regression tree?

An Example – `Hitters` data

- Year is the most important covariate in determining Salary – players with less experience earn lower salaries
- Given that a player is less experienced, the number of hits he made in the previous year seems to play little role for his salary
- Among players who have been in the major leagues for 5 or more years, the number of hits does affect salary, and players who made more hits have higher salaries
- Likely an over-simplification of the relationship between Salary and Years and Hits – but easier to interpret and a nice graphical representation available



Building a Regression Tree

- Let y_i be observed response variables and $x_{1i}, x_{2i}, \ldots, x_{pi}$ p predictor variables
- Building a regression tree:

1. Divide the predictor space (set of all possible values for $x_{1i}, \ldots, x_{pi}$) into J distinct and non-overlapping regions $R_1, \ldots, R_J$
2. For every observation that falls into region R_j, make the same prediction – the mean of the response values for the training observations in R_j

Building a Regression Tree

- Let y_i be observed response variables and $x_{1i}, x_{2i}, \ldots, x_{pi}$ p predictor variables
- Building a regression tree:

1. Divide the predictor space (set of all possible values for $x_{1i}, \ldots, x_{pi}$) into J distinct and non-overlapping regions $R_1, \ldots, R_J$
2. For every observation that falls into region R_j, make the same prediction – the mean of the response values for the training observations in R_j

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

Building a Regression Tree

- In theory, they could have any shape
- However, we choose to divide the predictor space into high-dimensional rectangles = **boxes** (for simplicity and ease of interpretation)

Building a Regression Tree

- In theory, they could have any shape
- However, we choose to divide the predictor space into high-dimensional rectangles = **boxes** (for simplicity and ease of interpretation)

Goal

Find boxes $R_1, \ldots, R_J$ that minimise the **Residual Sum of Squares**

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

where $\hat{y}_{R_j}$ is the mean response for the training observations within box R_j

Building a Regression Tree

- This is computationally infeasible when considering every possible partition of the feature space
- Hence, we use a **top-down**, **greedy** approach, known as **recursive binary splitting**
- Each split results into 2 branches further down the tree
- It is greedy because the **best** split is made at each step
- We select predictor x_j and the cutpoint s such that splitting the predictor space into regions $\{x|x_j < s\}$ and $\{x|x_j \geq s\}$ leads to the greatest possible reduction in the RSS

Building a Regression Tree

- For any j and s, define the pair of half-planes $R_1(j,s) = \{x|x_j < s\}$ and $R_2(j,s) = \{x|x_j \geq s\}$ and find j and s that minimise

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Building a Regression Tree

- For any j and s, define the pair of half-planes $R_1(j,s) = \{x|x_j < s\}$ and $R_2(j,s) = \{x|x_j \geq s\}$ and find j and s that minimise

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

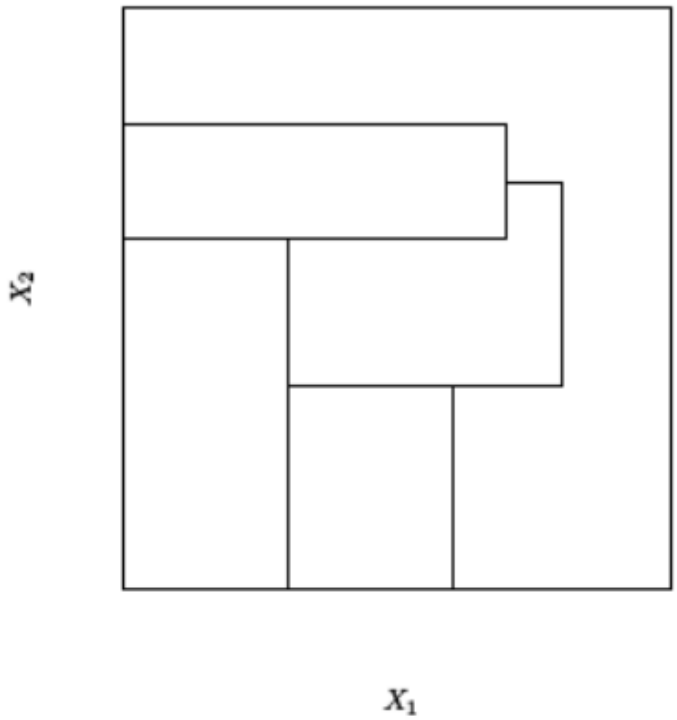
- Basically:
  (i) pick $j = 1$ and find s_1, ..., pick $j = p$ and find s_p
  (ii) make the split at s_j such that it corresponds to the minimum RSS

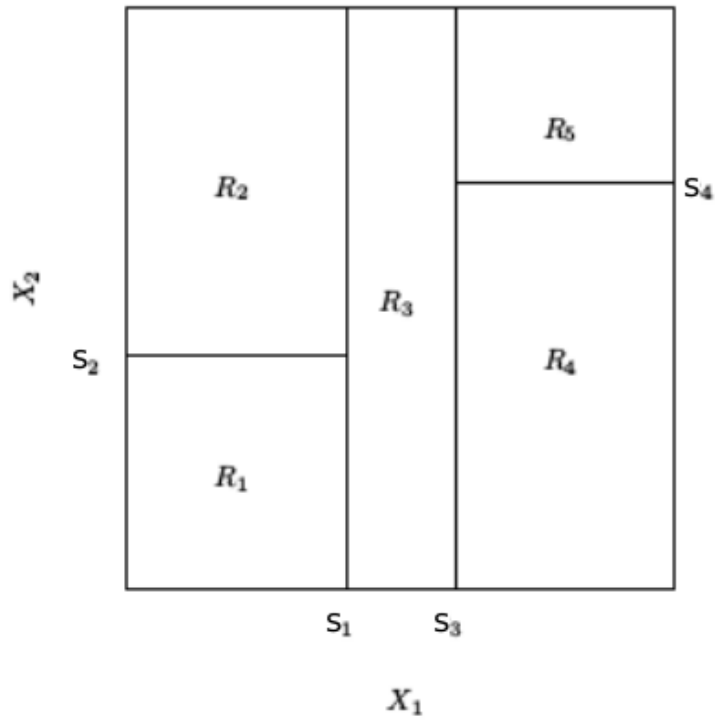
Building a Regression Tree

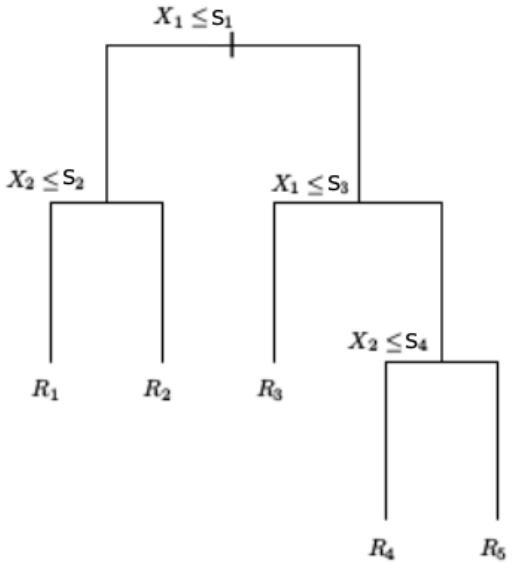
- For any j and s, define the pair of half-planes $R_1(j,s) = \{x|x_j < s\}$ and $R_2(j,s) = \{x|x_j \geq s\}$ and find j and s that minimise

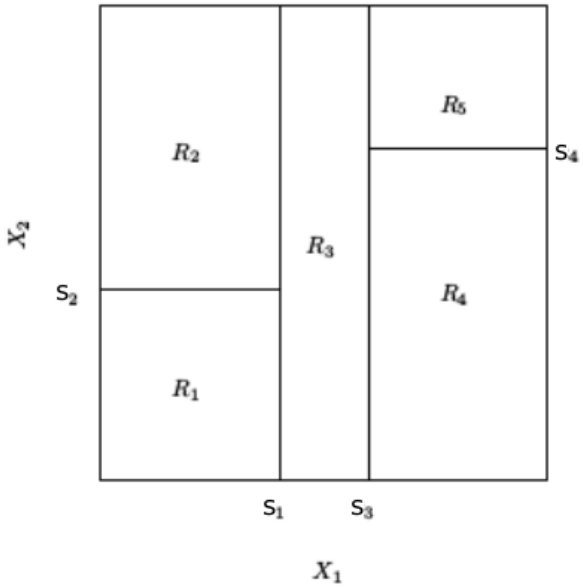
$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

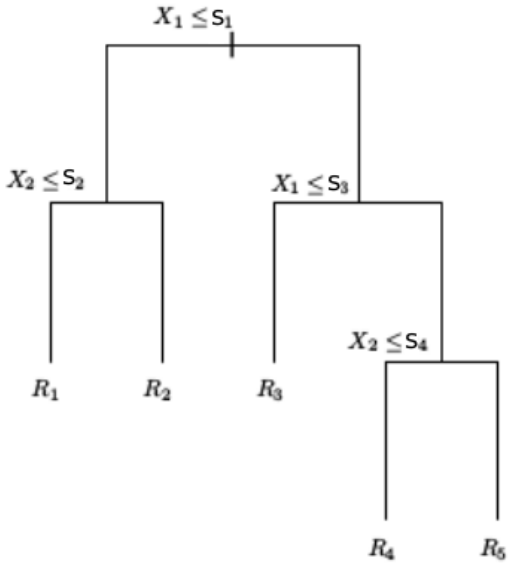
- Basically:
  (i) pick $j = 1$ and find s_1, ..., pick $j = p$ and find s_p
  (ii) make the split at s_j such that it corresponds to the minimum RSS
- We repeat the process for each resulting region, splitting the data further to minimise the RSS within each one of them until we reach a stopping criterion, e.g. until no region contains more than say 5 observations
- If the predictor variable is categorical, the split is made considering all possible combinations of levels

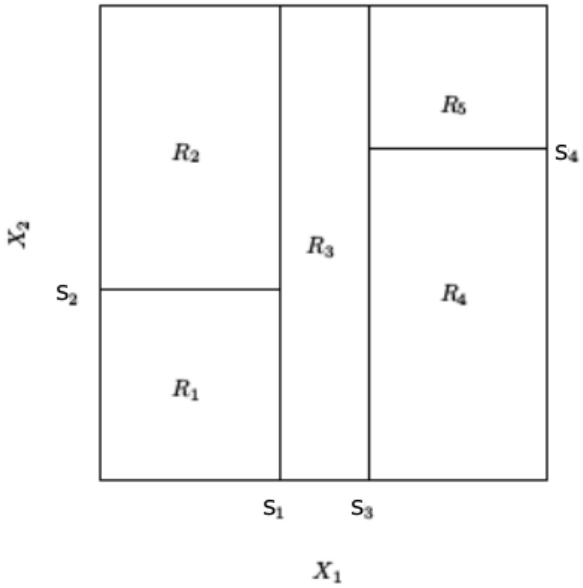

X_2
X_1

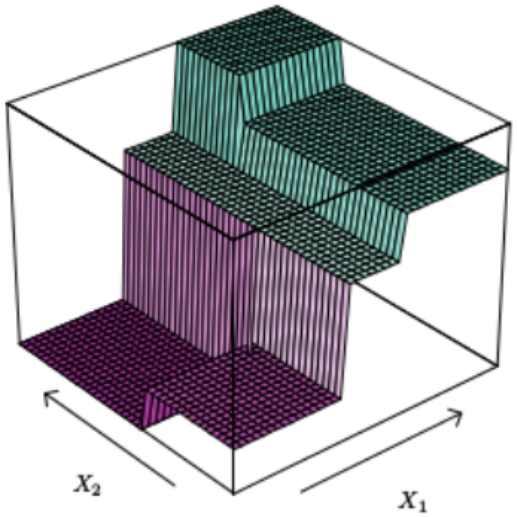

R_1
R_2
R_3
R_4
R_5
S_1
S_2
S_3
S_4
X_2
X_1


$X_1 \leq s_1$
$X_2 \leq s_2$
$X_1 \leq s_3$
$X_2 \leq s_4$
R_1
R_2
R_3
R_4
R_5


R_2
R_5
s_4
R_3
X_2
s_2
R_4
R_1
s_1
s_3
X_1


$X_1 \leq s_1$
$X_2 \leq s_2$
$X_1 \leq s_3$
$X_2 \leq s_4$
R_1
R_2
R_3
R_4
R_5


R_1
R_2
R_3
R_4
R_5
s_1
s_2
s_3
s_4
X_1
X_2


X_2
X_1

Regression Trees vs. Linear Models

- **Linear model**: $\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_{ji}$
- **Regression tree**: $\hat{y}_i = \sum_{m=1}^{M} \hat{c}_m . I(x_i \in R_m), \quad \hat{c}_m = \frac{1}{n_m} \sum_{x_i \in R_m} y_i$

Tree Pruning

- The resulting tree might be too complex
- A smaller tree may lead to better interpretation at the cost of a little bias
- Alternative: grow a very large tree T_0, then prune it back to obtain a subtree, selected so that it leads to the lowest error rate, using cross-validation

Tree Pruning

- The resulting tree might be too complex
- A smaller tree may lead to better interpretation at the cost of a little bias
- Alternative: grow a very large tree T_0, then prune it back to obtain a subtree, selected so that it leads to the lowest error rate, using cross-validation

Cost complexity pruning (a.k.a. weakest link pruning)

- Consider a sequence of trees indexed by a tuning parameter $\alpha \geq 0$. Then, for each value of α, there is a subtree $T \subset T_0$ such that

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

  is as small as possible; R_m is the subset corresponding to the m-th terminal node and $|T|$ is the number of terminal notes of tree T
- $\alpha = 0$ corresponds to tree T_0; as α grows, more complex trees are penalized

Cost Complexity Pruning

$$\alpha_0 = 0$$

Build T_0

Cost Complexity Pruning

$\alpha_0 = 0$

Build T_0

$$\begin{aligned} \alpha_1 &\equiv \text{subtree } T_1 \\ &\vdots \\ \alpha_q &\equiv \text{subtree } T_q \end{aligned}$$

$$\alpha_q > \alpha_{q-1} > \ldots > \alpha_1 > \alpha_0 = 0$$

Cost Complexity Pruning

$\alpha_0 = 0$

Build T_0

$$\begin{aligned} \alpha_1 &\equiv \text{subtree } T_1 \\ &\vdots \\ \alpha_q &\equiv \text{subtree } T_q \\ \alpha_q > \alpha_{q-1} > \ldots > \alpha_1 > \alpha_0 &= 0 \end{aligned}$$

K-fold cross-validation

Split the data into K subsets

$k = 1$

$$\begin{aligned} \alpha_1 &\equiv T_1^{(1)} \\ &\vdots \\ \alpha_q &\equiv T_q^{(1)} \end{aligned}$$

$\ldots$

$k = K$

$$\begin{aligned} \alpha_1 &\equiv T_1^{(K)} \\ &\vdots \\ \alpha_q &\equiv T_q^{(K)} \end{aligned}$$

removing observations from subset k

Cost Complexity Pruning

$\alpha_0 = 0$

Build T_0

$$\begin{aligned} \alpha_1 &\equiv \text{subtree } T_1 \\ &\vdots \\ \alpha_q &\equiv \text{subtree } T_q \\ \alpha_q > \alpha_{q-1} > \ldots > \alpha_1 > \alpha_0 &= 0 \end{aligned}$$

K-fold cross-validation

Split the data into K subsets

$$\begin{array}{ccc} k = 1 & & k = K \\ \alpha_1 \equiv T_1^{(1)} & \cdots & \alpha_1 \equiv T_1^{(K)} \\ \vdots & & \vdots \\ \alpha_q \equiv T_q^{(1)} & & \alpha_q \equiv T_q^{(K)} \end{array}$$

removing observations from subset k

For each subtree obtain the

$MSE_\alpha = \frac{1}{n_k} \sum_{i=1}^{n_k} (y_i - \hat{y}_i)^2$

$$\begin{array}{ccc} k = 1 & & k = K \\ MSE_{\alpha_1}^{(1)} & \cdots & MSE_{\alpha_1}^{(K)} \\ \vdots & & \vdots \\ MSE_{\alpha_q}^{(1)} & & MSE_{\alpha_q}^{(K)} \end{array}$$

Cost Complexity Pruning

$\alpha_0 = 0$

Build T_0

$$\begin{aligned} \alpha_1 &\equiv \text{subtree } T_1 \\ &\vdots \\ \alpha_q &\equiv \text{subtree } T_q \\ \alpha_q > \alpha_{q-1} > \ldots > \alpha_1 > \alpha_0 &= 0 \end{aligned}$$

K-fold cross-validation

Split the data into K subsets

$$\begin{array}{ccc|c|ccc} k & = & 1 & & k & = & K \\ \alpha_1 & \equiv & T_1^{(1)} & \cdots & \alpha_1 & \equiv & T_1^{(K)} \\ \vdots & & & & \vdots & & \\ \alpha_q & \equiv & T_q^{(1)} & & \alpha_q & \equiv & T_q^{(K)} \end{array}$$

removing observations from subset k

For each subtree obtain the

$MSE_\alpha = \frac{1}{n_k} \sum_{i=1}^{n_k} (y_i - \hat{y}_i)^2$

$$\begin{array}{ccc|c|ccc} k = 1 & & & & k = K & & \\ MSE_{\alpha_1}^{(1)} & & & \cdots & MSE_{\alpha_1}^{(K)} & & \\ \vdots & & & & \vdots & & \\ MSE_{\alpha_q}^{(1)} & & & & MSE_{\alpha_q}^{(K)} & & \end{array}$$

Average the MSE_αs over the k

Use the tree corresponding to the α that minimized the average MSE_α

Classification Trees

- Qualitative/categorical responses
- For regression trees, the predicted response is the group mean
- For classification trees, we predict that each observation belongs to the most occurring class of training observations in the region to which it belongs
- We also use recursive binary splitting, just as in the regression tree construction algorithm, but we cannot use the RSS as a criterion for making the splits
- Within possible alternatives, we may use the Gini index or the Cross-entropy measurement

Classification Trees

- **Gini index:** measure of total variance across all v classes

$$G = \sum_{v=1}^{V} \hat{p}_{mv}(1 - \hat{p}_{mv})$$

  $G \to 0$ as $\hat{p}_{mv} \to 0$ or $\hat{p}_{mv} \to 1$

- $\hat{p}_{mv}$ is the proportion of observations in the m-th region that belong to class v
- Measure of *node purity*, as a small G indicates that the node contains predominantly observations from a single class

Classification Trees

- **Gini index:** measure of total variance across all v classes

$$G = \sum_{v=1}^{V} \hat{p}_{mv}(1 - \hat{p}_{mv})$$

  $G \to 0$ as $\hat{p}_{mv} \to 0$ or $\hat{p}_{mv} \to 1$

- $\hat{p}_{mv}$ is the proportion of observations in the m-th region that belong to class v
- Measure of *node purity*, as a small G indicates that the node contains predominantly observations from a single class
- **Cross-entropy:**

$$D = -\sum_{v=1}^{V} \hat{p}_{mv} \log \hat{p}_{mv}$$

  $D \geq 0$, because $\hat{p}_{mv} \in [0, 1]$

- Again, $D \to 0$ as $\hat{p}_{mv} \to 0$ or $\hat{p}_{mv} \to 1$
- G and D are similar numerically

Gini / Entropy for $V = 2$ categories

Classification Trees: Example

- Spam dataset
- Dataset collected at the Hewlett-Packard Labs, that classifies 4601 e-mails as spam or non-spam
- 57 variables indicating the frequency of certain words/numbers and characters in the e-mail
- Goal: fit a predictive model that allows for the identification of spam e-mails

Spam dataset

- We will split the dataset into a training set and a validation set

```
data(spam, package="kernlab")

set.seed(1234)
select <- sample(1:nrow(spam), 1000)
spam_training <- spam[-select,]
spam_validation <- spam[select,]

spam[1:5, c(1:5, 53, 57, 58)]
```

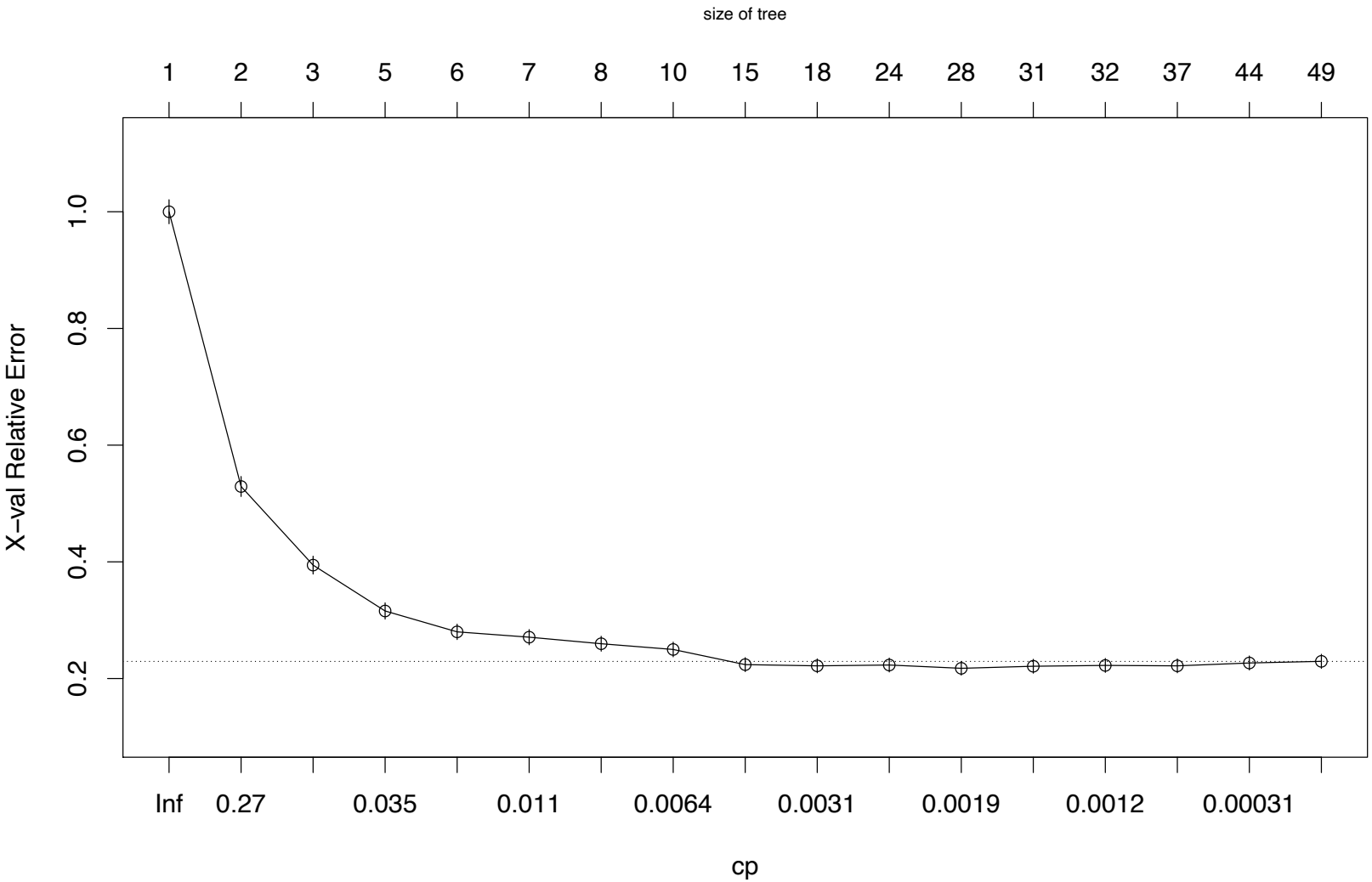
```
##   make address  all num3d  our charDollar capitalTotal type
## 1 0.00    0.64 0.64     0 0.32      0.000          278 spam
## 2 0.21    0.28 0.50     0 0.14      0.180         1028 spam
## 3 0.06    0.00 0.71     0 1.23      0.184         2259 spam
## 4 0.00    0.00 0.00     0 0.63      0.000          191 spam
## 5 0.00    0.00 0.00     0 0.63      0.000          191 spam
```

Spam dataset

- Fitting the classification tree

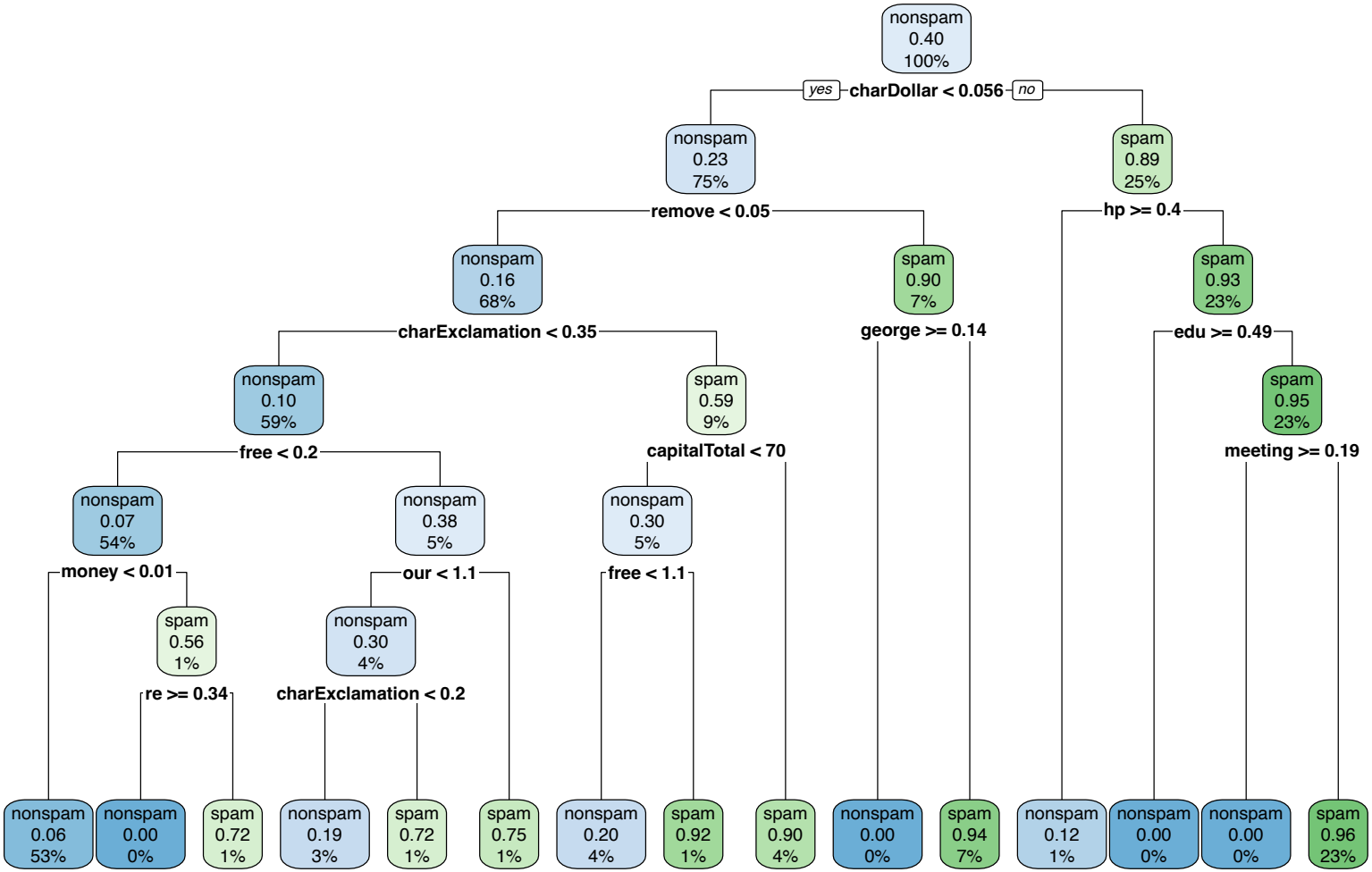
```
set.seed(123)
treeSpam <- rpart(type ~ ., data=spam_training, cp = 0)

plotcp(treeSpam)
```

Spam dataset



Spam dataset

```
treeSpam2 <- prune(treeSpam, cp = .0045)
rpart.plot(treeSpam2)
```



Spam dataset

- The probability of being spam associated with each e-mail of the validation dataset

```
predmat <- predict(treeSpam, newdata=spam_validation)
head(predmat)
```

```
##         nonspam        spam
## 1004 0.01104101 0.988958991
## 623  0.42857143 0.571428571
## 2693 0.99842767 0.001572327
## 934  0.02068966 0.979310345
## 4496 0.86666667 0.133333333
## 2948 0.96349413 0.036505867
```

```
pred <- predict(treeSpam, newdata=spam_validation, type='class')
```

- An e-mail will be classified as "spam" if the probability is greater than 0.5

Spam dataset

- Confusion table

```
ctable <- table(pred, spam_validation$type)
ctable
```

```
##
## pred      nonspam spam
##   nonspam     570   44
##   spam         40  346
```

- Estimated sensitivity – $P(\text{classify as spam}|\text{it is spam})$ and specificity – P(classify as non-spam|it is not spam)

```
sens <- ctable[2,2]/sum(ctable[,2]); sens
```

```
## [1] 0.8871795
```

```
espec <- ctable[1,1]/sum(ctable[,1]); espec
```

```
## [1] 0.9344262
```

Spam dataset

- Overall error rate:

```
err <- (ctable[1,2] + ctable[2,1])/sum(ctable); err
```

```
## [1] 0.084
```

Spam dataset

- Here we are using $p = 0.50$ as a threshold for classifying an e-mail as spam. If discarding a non-spam e-mail is more damaging than not discarding an actual spam e-mail, a larger threshold for p would be more appropriate, i.e. $p = 0.90$

```
pred09 <- ifelse(predmat[,2] > 0.9, 'spam', 'nonspam')
ctable2 <- table(pred09, spam_validation$type)
ctable2
```

```
##
## pred09    nonspam spam
##   nonspam     591   77
##   spam         19  313
```

```
ctable2[2,2]/sum(ctable2[,2])
```

```
## [1] 0.8025641
```

```
ctable2[1,1]/sum(ctable2[,1])
```

```
## [1] 0.9688525
```

Decision Trees

Advantages / Disadvantages

- decision trees are easy to explain
- they closely mirror human decision making
- graphical displays are available
- decision trees generally don't have the same level of predictive accuracy as other regression/classification approaches
- trees are highly unstable; small changes in the data may generate considerably different fits

Extensions

- Improving predictive performance: bootstrap aggregation or **Bagging**, **Random forests**, **Boosting**
- There are extensions for count and survival data
- QUEST, GUIDE, CTree, . . . – recursive partitioning algorithms with stopping rules based on p-values
- Bayesian CART (B-CART), Bayesian Additive Regression Trees (BART) and many other extensions

Random forests

- The decision trees suffer from high variance (ie. if we reperatedly fitted the model to different training sets we expect a large variation in results).
- Bootstrap aggregation, of **bagging** is a general purpose method to reduce variance in statistical learning methods.
- It works on this premise: for n independent random variables (observations) with variance σ^2, the variance of their mean is σ^2/n.
- So to reduce the variance and increase prediction accuracy of a method, use many training sets from a population, build an independent model on each one and average the resulting predictions.
- Often, however, we do not have access to multiple training sets. Instead we **bootstrap**, i.e. take repeated samples (with replacement) from a single training set.

Random forests

- So: obtain a number of bootstrapped training sets, build a tree on each one (without pruning) and average the resulting predictions for each observation.
- Whereas for regression trees, the average of a prediction is straightforward, in classification trees we can use majority vote.
- How many bootstrap training sets to use? Increasing this quantity will not lead to overfitting, so the more, the merrier.
- Conveniently, it turns out that it is straightforward to estimate the error rate of bagged models without crossvalidation. It can be shown that, on average, each bootstrapped training set will leave out one third of the observations. We call these out of bag (OOB) observations.

Random forests

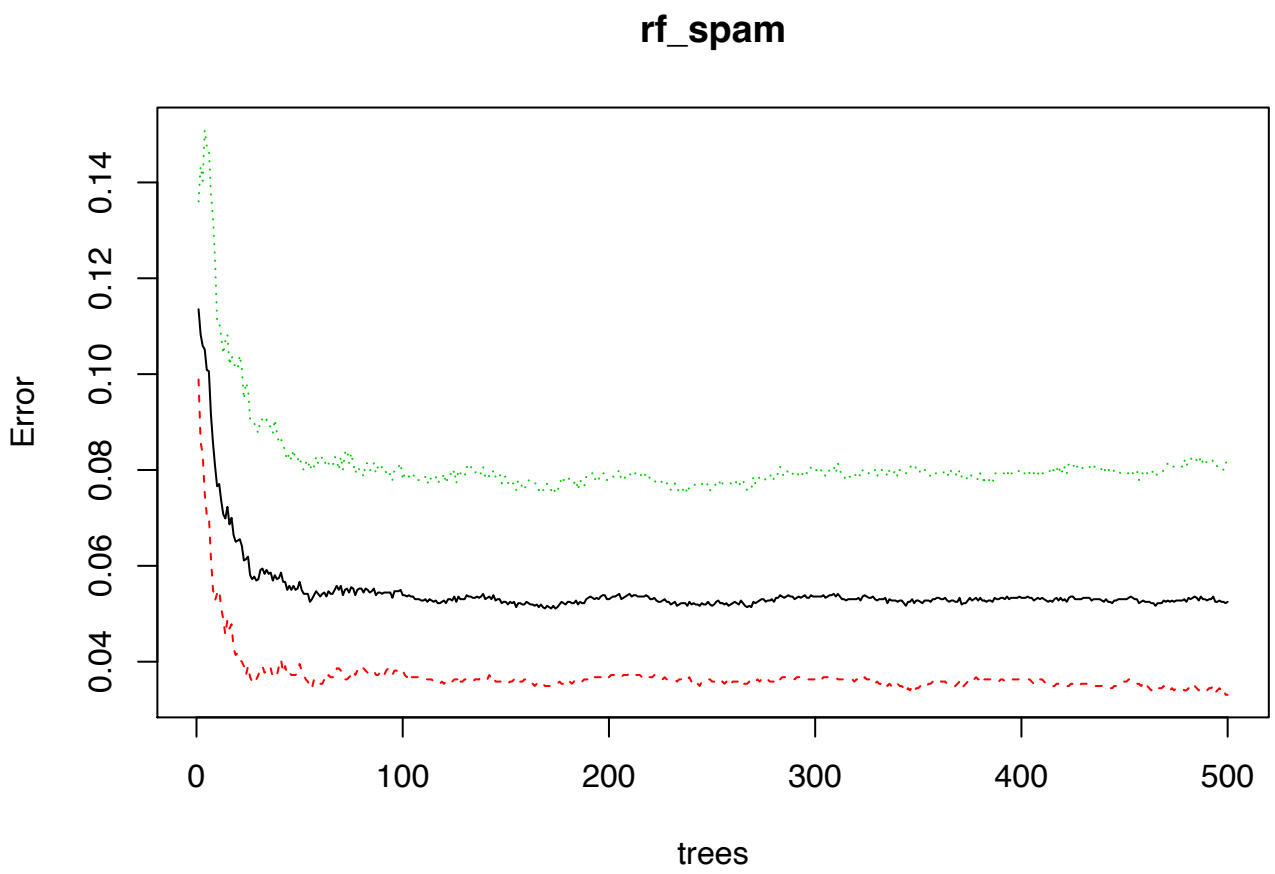
- Thus, each observation can be predicted from the trees where it was OOB and the average of this can be computed. The overall OOB MSE or classification error can be estimated for the entire dataset.
- This is a valid estimate of the test error of the bagged model, since the prediction for each observation is made using only trees that were fit without that observation.

Random forests

- The advantage of bagging is improved accuracy, but the disadvantage is loss of interpretability. However we can estimate the importance of each predictor in the resulting model by looking at the total amount that the SSE (or the Gini index for classification) decreases due to splits over that predictor averaged over all the trees. A large value indicates an important predictor.
- **Random forests** are based on the bagging principle, i.e. a number of trees are built on bootstrapped training samples. But when building a tree, each time a split is considered, only a random sample of predictors is considered (usually $\sqrt{p}$). Each split takes a new sample of predictors for consideration. This has the effect of **decorrelating** the trees by increasing the variablity of the obtained trees.

Random forests - Spam dataset

```
library(randomForest)
set.seed(2019)
rf_spam <- randomForest(type ~ ., data = spam_training)
plot(rf_spam)
```



Random forests - Spam dataset

- The probability of being spam associated with each e-mail of the validation dataset

```
predmat <- predict(rf_spam, newdata=spam_validation)
pred <- predict(rf_spam, newdata=spam_validation, type='class')
```

- Again, an e-mail will be classified as "spam" if the probability is greater than 0.5

Random forests - Spam dataset

- Confusion table

```
ctable <- table(pred, spam_validation$type)
ctable
```

```
##
## pred      nonspam spam
##   nonspam     595   39
##   spam         15  351
```

- Estimated sensitivity – $P(\text{classify as spam}|\text{it is spam})$ and specificity – P(classify as non-spam|it is not spam)

```
sens <- ctable[2,2]/sum(ctable[,2]); sens
```

```
## [1] 0.9
```

```
espec <- ctable[1,1]/sum(ctable[,1]); espec
```

```
## [1] 0.9754098
```

Random forests - Spam dataset

- Overall error rate:

```
err <- (ctable[1,2] + ctable[2,1])/sum(ctable); err
```

```
## [1] 0.054
```

Random forests - Spam dataset

- We now look at variable importance

```
varImpPlot(rf_spam)
```