Regression Trees

Trevor Hastie and Robert Tibshirani

Here, I am adapting part of the lab associated with Chapter 8 of the textbook.

The `tree` library is used to construct classification and regression trees.

```
#install.packages("tree")
library(tree)
library(ISLR2)
```

Fitting Regression Trees

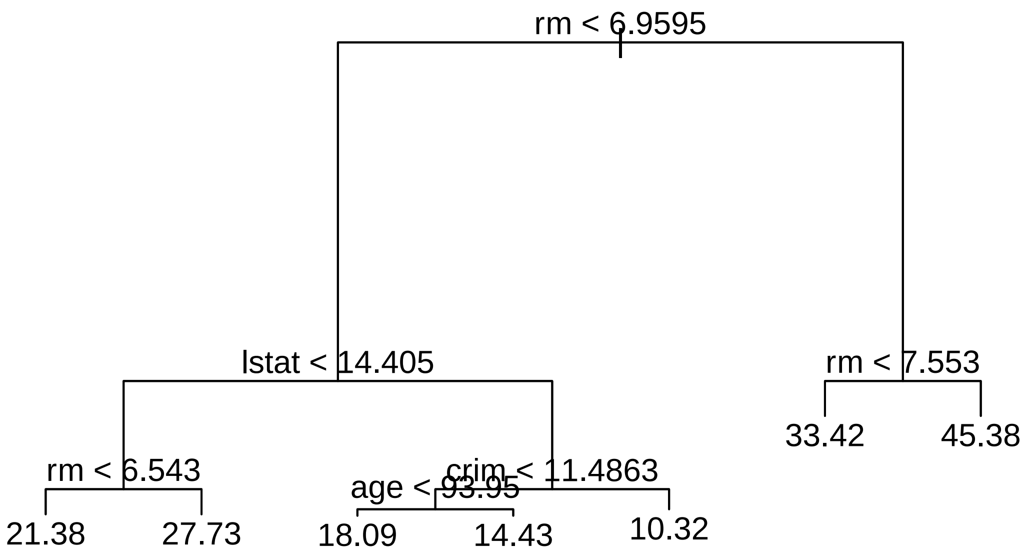
Here we fit a regression tree to the `Boston` data set. First, we create a training set, and fit the tree to the training data.

```
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston) / 2)
tree.boston <- tree(medv ~ ., Boston, subset = train)
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.38 = 2555 / 246
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

Notice that the output of `summary()` indicates that only four of the variables have been used in constructing the tree. In the context of a regression tree, the deviance is simply the sum of squared errors for the tree. We now plot the tree.

```
plot(tree.boston)
text(tree.boston, pretty = 0)
```



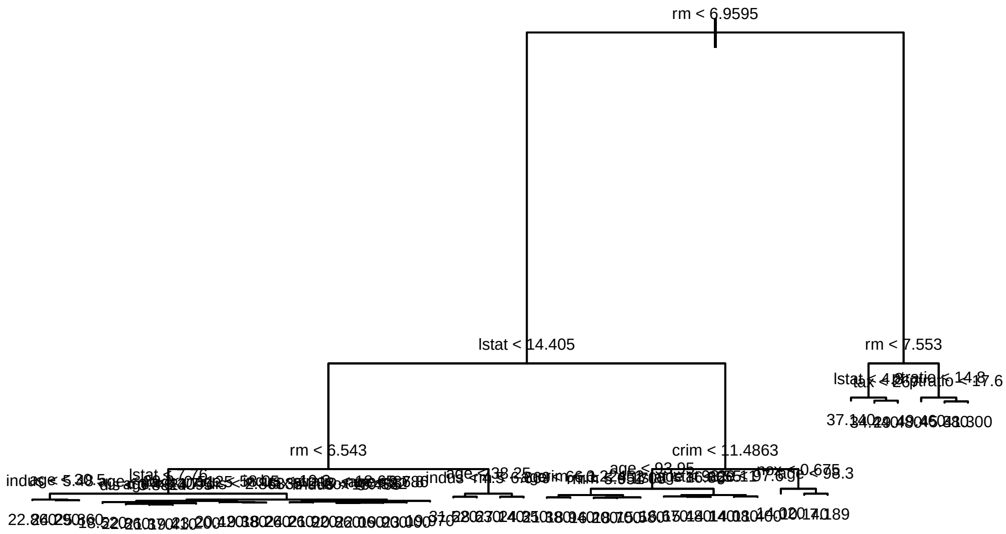
The variable `lstat` measures the percentage of individuals with lower socioeconomic status, while the variable `rm` corresponds to the average number of rooms. The tree indicates that larger values of `rm`, or lower values of `lstat`, correspond to more expensive houses. For example, the tree predicts a median house price of 45,400 for homes in census tracts in which `rm >= 7.553`.

It is worth noting that we could have fit a much bigger tree, by passing `control = tree.control(nobs = length(train), mindev = 0)` into the `tree()` function. What if we do that?

```
tree.boston.big <- tree(medv ~ ., Boston, subset = train,
                        control = tree.control(nobs = length(train),
                                               mindev = 0))
summary(tree.boston.big)
```

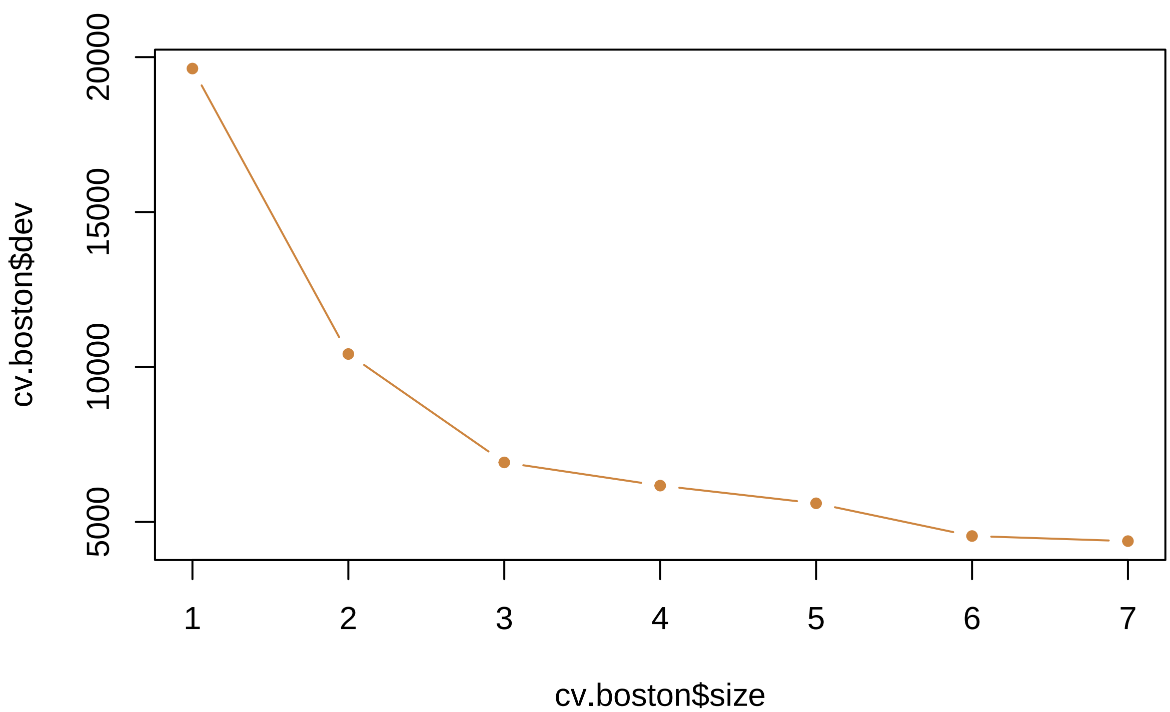
```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train, control = tree.control(nobs = length(train),
##     mindev = 0))
## Variables actually used in tree construction:
## [1] "rm"      "lstat"   "indus"   "age"     "nox"     "dis"     "ptratio"
## [8] "tax"     "crim"
## Number of terminal nodes:  41
## Residual mean deviance:  5.542 = 1175 / 212
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -8.140  -1.200   0.000   0.000   1.087  12.860
```

```
plot(tree.boston.big)
text(tree.boston.big, pretty = 0, cex=0.5)
```



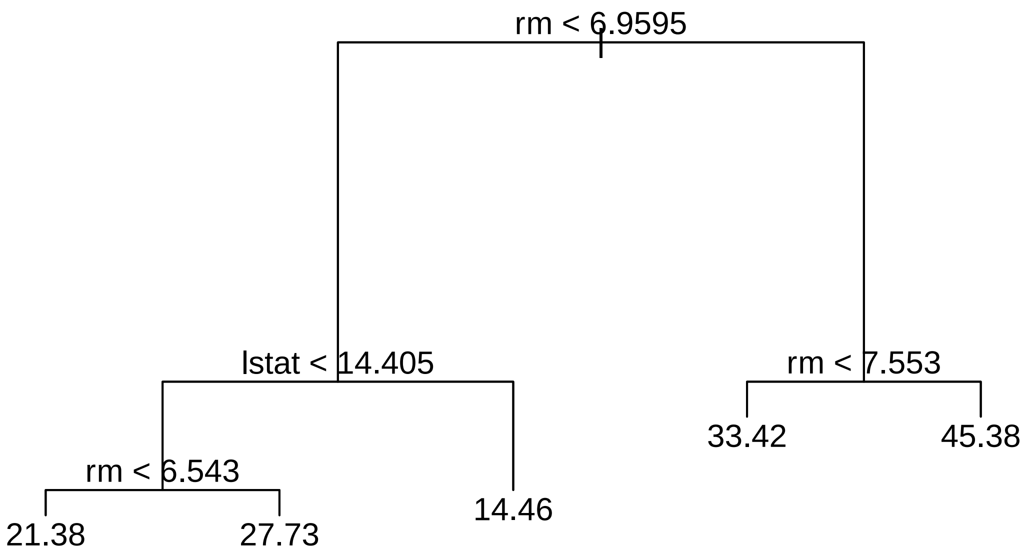
Now we use the `cv.tree()` function to see whether pruning the tree will improve performance.

```
cv.boston <- cv.tree(tree.boston)
plot(cv.boston$size, cv.boston$dev, type = "b",
     col="peru", pch=20)
```



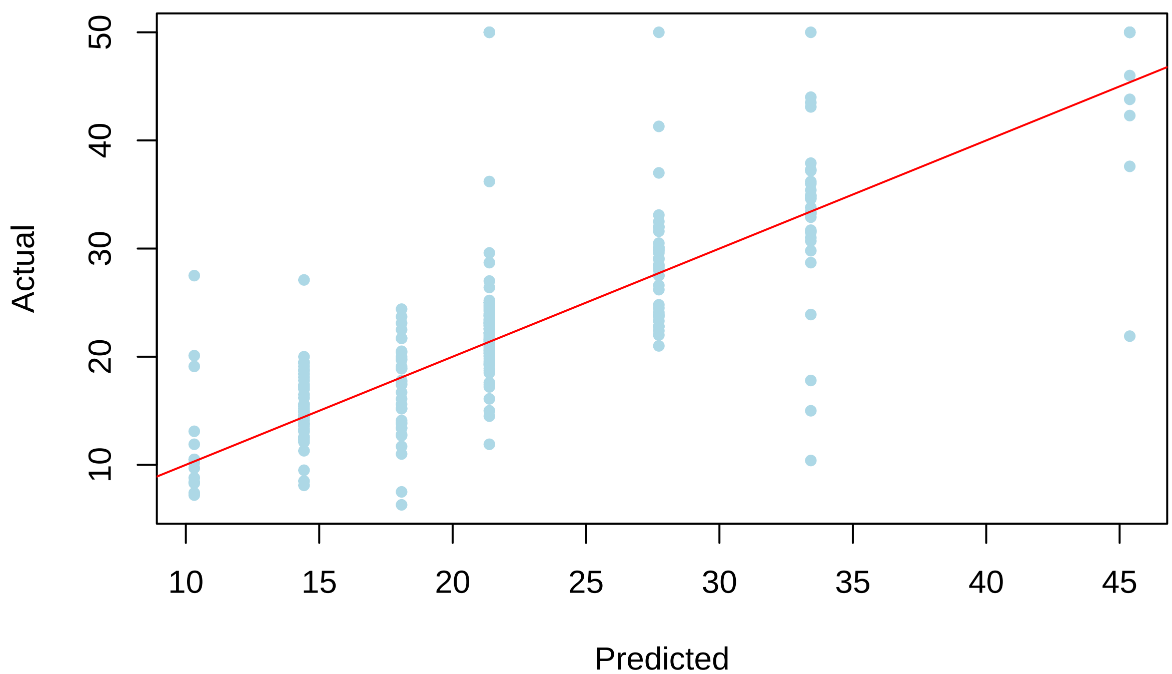
In this case, the most complex tree under consideration is selected by cross-validation. However, if we wish to prune the tree, we could do so as follows, using the `prune.tree()` function:

```
prune.boston <- prune.tree(tree.boston, best = 5)
plot(prune.boston)
text(prune.boston, pretty = 0)
```



In keeping with the cross-validation results, we use the **unpruned** tree to make predictions on the test set.

```
y.hat <- predict(tree.boston, newdata = Boston[-train, ])
boston.test <- Boston[-train, "medv"]
plot(y.hat, boston.test, col="lightblue", pch=20,
     xlab="Predicted", ylab="Actual")
abline(0, 1, col="red")
```




```
mean((y.hat - boston.test)^2)
```

```
## [1] 35.28688
```

In other words, the test set MSE associated with the regression tree is 35.29. The square root of the MSE is therefore around 5.941, indicating that this model leads to test predictions that are (on average) within approximately 5,941 of the true median home value for the census tract.