Assignment #9

Brody Vogel

Preparation

```
require("ISLR")
```

```
## Loading required package: ISLR
```

```
require("MASS")
```

```
## Loading required package: MASS
```

```
require("e1071")
```

```
## Loading required package: e1071
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.4.4
```

```
library(gbm)
```

```
## Loading required package: survival
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
```

Basic Data Description

We used the *airfoil* data from the UCI Machine Learniing Repository. From the sparse description on the site, it looks like the goal was to predict the sound made by aircraft wing blades with varying attributes.

The features of the dataset are: Frequency, Angle of Attack, Chord Length, Free-Stream Velocity, and Suction-Side Displacement. There's very little to go on as far as descriptions, but it looks like Angle of Attack and Chord Length have to do with the angled dimensions and length of the wing, while Frequency and Free-Stream Velocity refer vaguely to the speed at which the aircraft is moving; Suction-Side Displacement seems to have something to do with the complexities of the wing's aerodynamic structure.

The response that we're trying to predict is Scaled Sound Pressure Level. If I had to guess, I think I'd assume the people collecting the data wanted to test the sound made by stealth fighter jets based on the differences in their wings. I would think that a good fighter jet would be a quiet fighter jet, and so they wanted to see what they could do to make the aircrafts quieter.

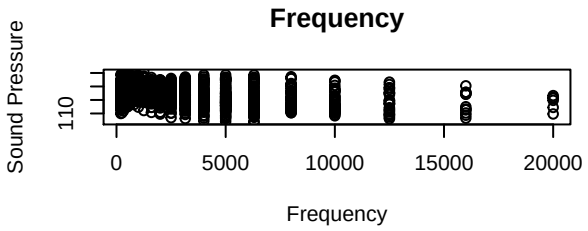
Data Collection and "Cleaning"

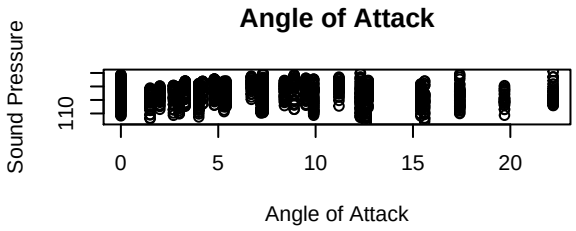
The dataset is clean (as far as we could tell without knowing what some of the predictors referred to). There are no missing values. The eyetest doesn't highlight any outliers in the scatterplots, either.

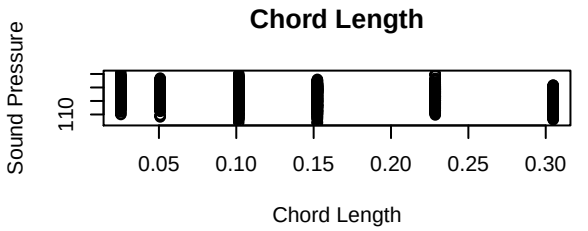
```
load("/Users/brodyvogel/Downloads/airfoil.RData")
# no missing values
sum(is.na(airfoil))
```

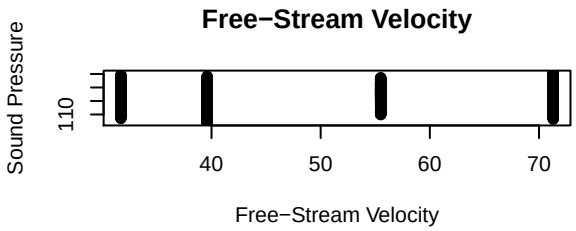
```
## [1] 0
```

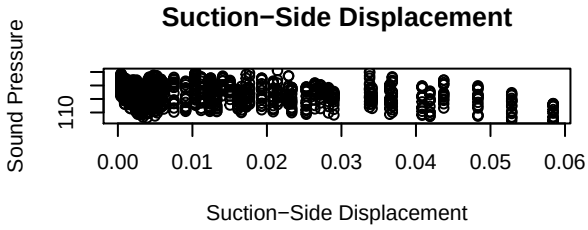
```
#head(airfoil)

par(mfrow = c(3, 2))
plot(airfoil$Frequency, airfoil$Pressure, main = 'Frequency', ylab = 'Sound Pressure', xlab = 'Frequency
plot(airfoil$AngleOfAttack, airfoil$Pressure, main = 'Angle of Attack', ylab = 'Sound Pressure', xlab =
plot(airfoil$ChordLength, airfoil$Pressure, main = 'Chord Length', ylab = 'Sound Pressure', xlab = 'Cho
plot(airfoil$FreeStreamVelocity, airfoil$Pressure, main = 'Free-Stream Velocity', ylab = 'Sound Pressur
plot(airfoil$Displacement, airfoil$Pressure, main = 'Suction-Side Displacement', ylab = 'Sound Pressure
```











Exploratory Data Analysis

From a quick correlation check, it seems that "AngleOfAttack" and "Displacement" are somewhat correlated (.75). That'll be something to look out for.

The boxplots below seem to tell us that Frequency has the most clear relationship with Sound Pressure: as the Frequency grows from 200 to ~1200, the Sound Pressure grows, too; after that, the Sound Pressure

decreases as the Frequency increases.

AngleOfAttack seems to have a similar relationship with Sound Pressure: at first, the two grow together, but after a certain point Sound Pressure decreases as Angle increases.

Sound Pressure looks like it increases with Velocity, although the effect seems to be small.

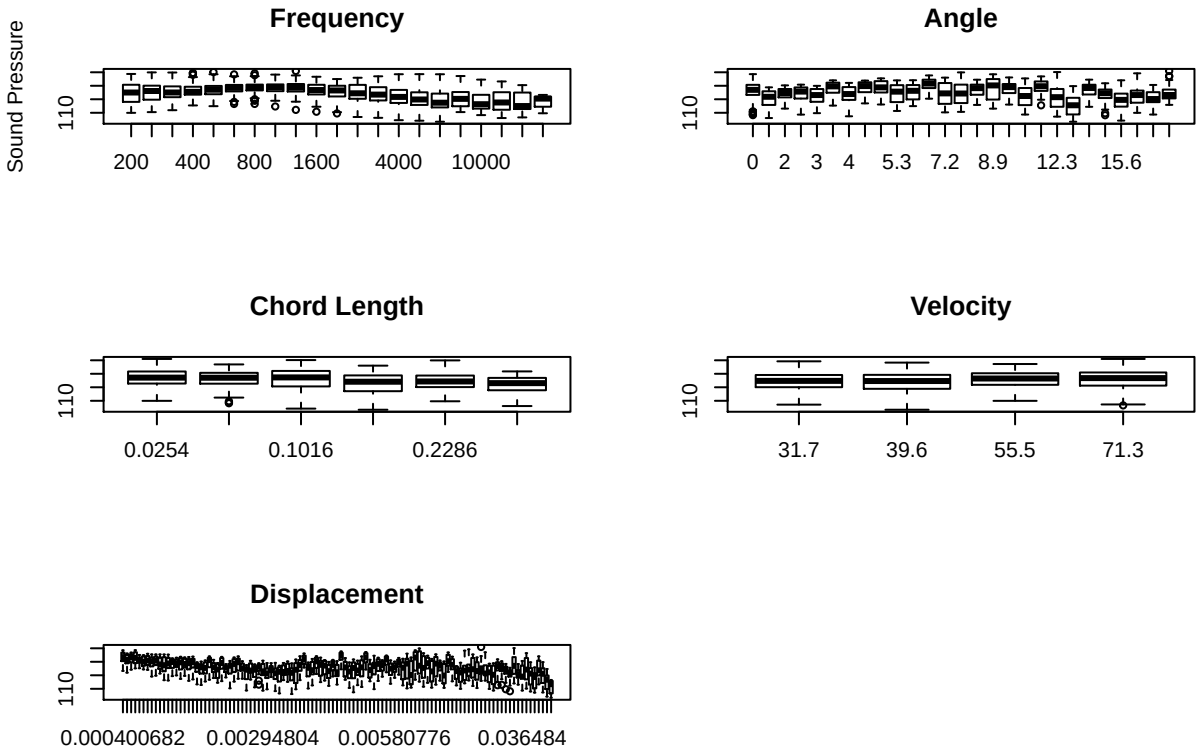
The relationship between Sound Pressure and the Chord Length and Displacement predictors is fairly ambiguous in this plot.

Again, if I had to guess, I'd say that the aircrafts get louder as they move faster, level off at some point, and then see their sound bounce around a bit unpredictably at the highest speeds.

```
cor(airfoil)
```

```
##                      Frequency AngleOfAttack  ChordLength
## Frequency           1.000000000   -0.27276454 -0.003660639
## AngleOfAttack      -0.272764536    1.00000000 -0.504868150
## ChordLength        -0.003660639   -0.50486815  1.000000000
## FreeStreamVelocity  0.133663831    0.05875957  0.003786629
## Displacement       -0.230107353    0.75339378 -0.220842431
## Pressure           -0.390711412   -0.15610753 -0.236161512
##                    FreeStreamVelocity Displacement   Pressure
## Frequency                 0.133663831 -0.230107353 -0.3907114
## AngleOfAttack             0.058759565  0.753393785 -0.1561075
## ChordLength               0.003786629 -0.220842431 -0.2361615
## FreeStreamVelocity        1.000000000 -0.003974013  0.1251028
## Displacement             -0.003974013  1.000000000 -0.3126695
## Pressure                  0.125102801 -0.312669506  1.0000000
```

```
#pairs(airfoil)
par(mfrow = c(3, 2))
boxplot(Pressure~Frequency, data = airfoil, main = "Frequency", ylab = 'Sound Pressure')
boxplot(Pressure~AngleOfAttack, data = airfoil, main = "Angle")
boxplot(Pressure~ChordLength, data = airfoil, main = "Chord Length")
boxplot(Pressure~FreeStreamVelocity, data = airfoil, main = "Velocity")
boxplot(Pressure~Displacement, data = airfoil, main = "Displacement")
```



Training and Test Sets

Our dataset isn't particularly large (1503 entries), so we split it roughly 50/50 between training and test sets.

```
train = sample(c(TRUE, FALSE), nrow(airfoil), rep = TRUE)
test = (!train)
training = airfoil[train, ]
testing = airfoil[test, ]
```

Baseline Linear Model

When we fit a basic linear model without interactions, it looked like each of the five predcitors are important (each has a p-value of essentially 0). However, the R^2 is only .52, so the basic model isn't doing a great job of explaining the variability in the data. Furthermore, when the model was tested, it produced a RMSE of about 5, which corresponds to an average error of something like 5%.

We remembered the apparent correlation between Displacement and AngleOfAttack, and added an interaction term between the two to a potential model. The interaction term was a useless feature (p-value = .63), and so we dropped it.

```
# it looks like each of the values is quite important, at least based on the p-values
                            # or training - results change very little
lin.fit <- lm(Pressure ~ ., data = airfoil)

summary(lin.fit)
```

```
##
## Call:
```

```
## lm(formula = Pressure ~ ., data = airfoil)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.480  -2.882  -0.209   3.152  16.064
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.328e+02  5.447e-01  243.87   <2e-16 ***
## Frequency          -1.282e-03  4.211e-05  -30.45   <2e-16 ***
## AngleOfAttack      -4.219e-01  3.890e-02  -10.85   <2e-16 ***
## ChordLength        -3.569e+01  1.630e+00  -21.89   <2e-16 ***
## FreeStreamVelocity  9.985e-02  8.132e-03   12.28   <2e-16 ***
## Displacement       -1.473e+02  1.501e+01   -9.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.809 on 1497 degrees of freedom
## Multiple R-squared:  0.5157, Adjusted R-squared:  0.5141
## F-statistic: 318.8 on 5 and 1497 DF,  p-value: < 2.2e-16
```

```
sqrt(mean((predict(lin.fit, testing) - testing$Pressure)^2))
```

```
## [1] 4.802886
```

```
#lin.fit2 <- lm(Pressure ~ . + Displacement:AngleOfAttack, data = airfoil)

#summary(lin.fit2)
```

Simple Decision Tree and Visualization

When we fit a simple decision tree, the method only used four predictors: "Frequency", "Displacement", "FreeStreamVelocity", and "ChordLength" to produce 18 terminal nodes. This suggests "AngleOfAttack" is less related to the Sound Pressure. Furthermore, the plotted tree uses "Frequency", "Displacement", and "FreeStreamVelocity" fairly frequently to make splits, but "ChordLength" only appears once. So, maybe, "ChordLength" is less related to Sound Pressure, too.

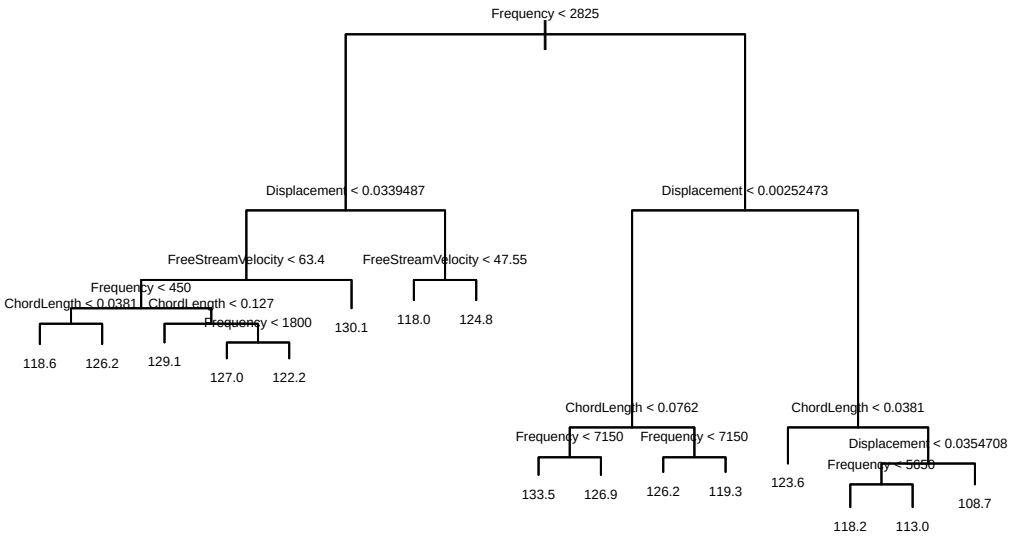
When used on the test data, the model produced a MSE of 19.05 (RMSE of 4.36). This is marginally better than the results from the simple model. Thus, using a basic tree gave us a little insight into the predictors that may be most important in the model, but didn't improve the performance of that model much over simple linear regression.

```
mytree <- tree(Pressure ~ ., data = training)
summary(mytree)
```

```
##
## Regression tree:
## tree(formula = Pressure ~ ., data = training)
## Variables actually used in tree construction:
## [1] "Frequency"          "Displacement"       "FreeStreamVelocity"
## [4] "ChordLength"
## Number of terminal nodes:  16
## Residual mean deviance:  16.6 = 12100 / 729
## Distribution of residuals:
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
```

```
## -17.570000  -2.723000  -0.001227   0.000000   2.680000  11.900000
```

```
plot(mytree)
text(mytree, pretty = 2, cex = .4)
```




```
MSE = sum((predict(mytree, newdata = testing) - testing$Pressure)^2)/nrow(testing)
MSE
```

```
## [1] 19.56287
```

```
sqrt(MSE)     #RMSE
```

```
## [1] 4.422993
```

Bagging

The results from our fitting of a bagged model were much better than those from the linear regression and simple decison tree models. According to the output, when using 500 trees, the model explained over 88 percent of the variability in training data. When it was applied to the test data, then, the bagged model produced a MSE of 4.63 (RMSE = 2.15). This is about 1/5 the error produced by the linear regression and simple decision tree models.

```
bag.fit <- randomForest(Pressure ~ ., data = training, mtry = 5, replace = TRUE)

bag.fit
```

```
##
## Call:
##  randomForest(formula = Pressure ~ ., data = training, mtry = 5,      replace = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 4.895397
##                     % Var explained: 89.86
```

```
MSE = sum((predict(bag.fit, newdata = testing) - testing$Pressure)^2)/nrow(testing)
```

```
MSE
```

```
## [1] 5.106931
```

```
sqrt(MSE)
```

```
## [1] 2.259852
```

Random Forest

We next tried to fit a random forest model with 3 random predictors chosen to build the tree at each step. Again with 500 trees, the output told us the random forest model captured about 87 percent of the variability in the training data. When then applied to the test data, the model produced a MSE of 5.07 (RMSE = 2.25). These results are pretty similar - although a bit worse - to what we got from bagging. This was expected, though: there were only five predictors in the airfoil dataset, and so a random selection of three will still likely produce something close to the optimal split. Since bagging is just a special case of random forest where the number of predictors chosen from at each step is the number of predictors in the dataset, we were basically performing the same analysis with our application of random forest as we were with bagging; this explains the similar results.

```
RF.fit <- randomForest(Pressure ~ ., data = training, mtry = 3, replace = TRUE)

RF.fit
```

```
##
## Call:
##  randomForest(formula = Pressure ~ ., data = training, mtry = 3,      replace = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 5.640703
##                     % Var explained: 88.31
```

```
MSE = sum((predict(RF.fit, newdata = testing) - testing$Pressure)^2)/nrow(testing)

MSE
```

```
## [1] 5.816186
```

```
sqrt(MSE)
```

```
## [1] 2.411677
```

Boosting

When we fit a boosted tree, we got a little better results. When we just used 500 trees and a maximal tree depth of 4, our results were basically the same as with bagging and random forest. When we pushed the number of trees to 10000 and the maximal tree depth to 20, though, we got a MSE of only 3.59 (RMSE = 1.89). These, then, were our best results.

```
# the results are about the same as that from the random forest or bagged models when using a reasonabl

boost.fit <- gbm(Pressure ~ ., data = training, n.trees = 10000, interaction.depth = 20)
```

```
## Distribution not specified, assuming gaussian ...
```

```
boost.fit
```

```
## gbm(formula = Pressure ~ ., data = training, n.trees = 10000,
##     interaction.depth = 20)
## A gradient boosted model with gaussian loss function.
## 10000 iterations were performed.
## There were 5 predictors of which 5 had non-zero influence.
```

```
MSE = sum((predict(boost.fit, newdata = testing, n.trees = 10000) - testing$Pressure)^2)/nrow(testing)

MSE
```

```
## [1] 4.13662
```

```
sqrt(MSE)
```

```
## [1] 2.033868
```

Conclusions

Ultimately, the best method (in terms of results) that we tried for fitting a model to the *airfoil* data was boosting with a large number of trees and maximal tree depth. When allowed to extend itself in this way, the method seemed to account for most of the relationship between the five predictors and the response, Sound Pressure.

We learned that the Frequency, Free-Stream Velocity, and Suction-Side Displacement were the most useful in predicting the Sound Pressure produced by the various aircraft wings. In particular, Sound Pressure seemed to grow with Frequency to a certain point, and then decrease as Frequency increased. I'm sure this is interesting from an aerodynamics perspective, but, seeing as I lack that perspective entirely, I found these results to be interesting more because I expected the Sound Pressure to increase monotonically with each of the predictors. As we've seen, though, that is not the case - there appear to be some modifications that can be made to make the aircrafts quieter. That in itself seems like a useful takeaway from this study.