Final Project

Tyler Chang

STA 135 Multivariate Analysis

Professor Xiaodong Li

10 March 2020

Final Project

Part 1: Multiple Linear Regression

Introduction

In this section, we will analyze the *Auto MPG* data set from the UC Irvine Machine Learning Repository. In particular, we will create a multiple linear regression model to measure the extent to which a car's miles per gallon (MPG) can be explained by other explanatory variables (i.e. displacement, horsepower, etc.).

Summary

The goal of this section is to construct a multiple linear regression model to observe any potential relationships MPG (Y) has with the independent variables: displacement (z_1), horsepower (z_2), weight (z_3), and acceleration (z_4). In other words, we are trying to model:

$$Y = \beta_0 + \beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3 + \beta_4 z_4 + \epsilon$$

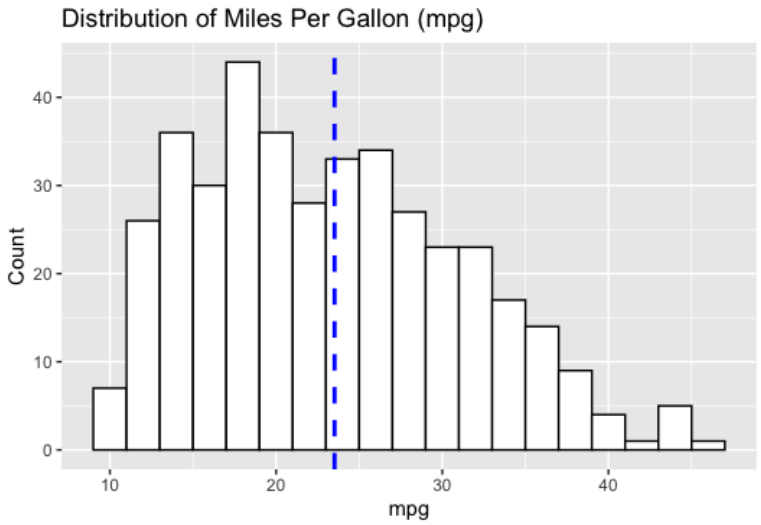
Summary Statistics

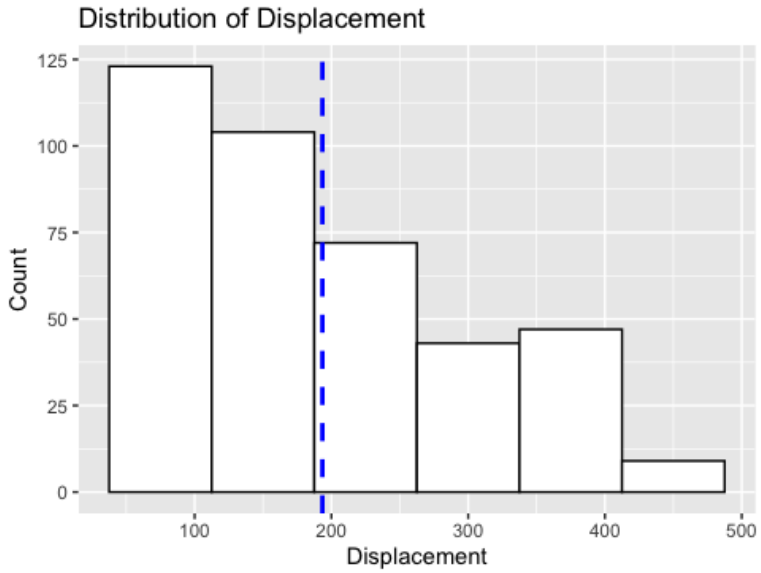
	Y	z_1	z_2	z_3	z_4
Min	9	68	1	1613	8
Q1	17.5	104.2	26	2224	13.82
Median	23	148.5	60.5	2804	15.5
Mean	23.51	193.4	51.39	2970	15.57
Q3	29	262	79	3608	17.18
Max	46.6	455	94	5140	24.8

Final Project

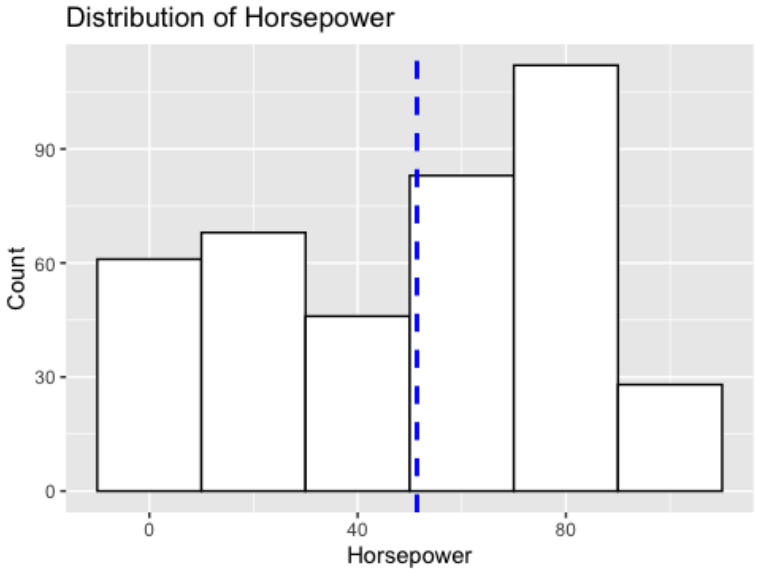
Plots/Distributions

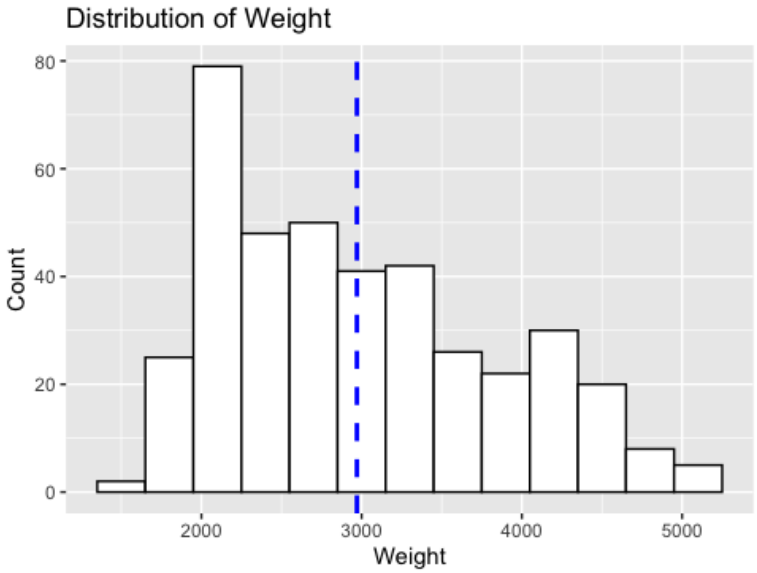
Plotted below are histograms of each of the variables in the model. The mean of each respective variable is marked with a blue dashed line.

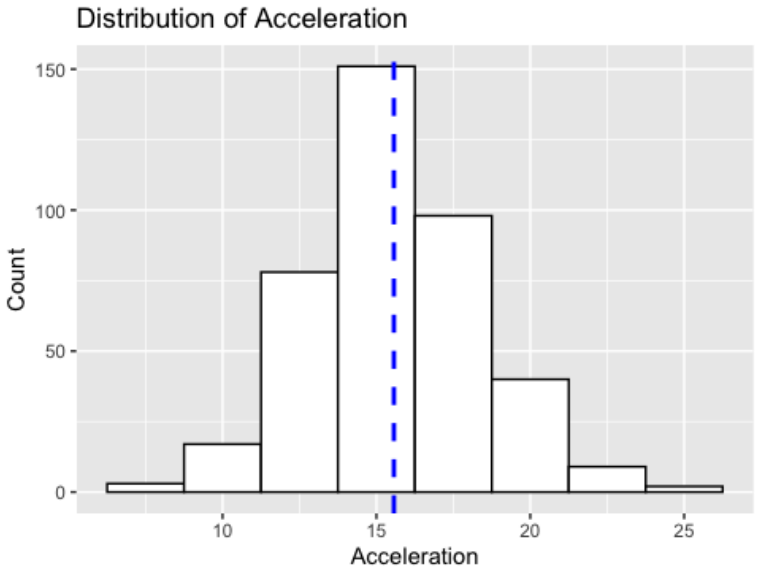




Final Project







Analysis

After using the least squares method to obtain the regression intercept and coefficients, the fitted model is as follows:

$$\hat{Y} = 40.883803 - 0.010629z_1 + 0.004777z_2 - 0.006140z_3 + 0.172216z_4$$

To observe the strength of the model, we calculate the coefficient of determination, namely R^2. After fitting the model, we observe $R^2 = .7006485$. This indicates that there is a moderately strong, positive linear relationship between the response, Y, and the independent variables z_i.

Alternatively, we can interpret R^2 as the proportion of variation explained by the data ($R^2 = \frac{ESS}{TSS}$). In other words, approximately 70.06% of variation can be explained by the data.

F Test

In this section we will test the following:

$$H_0 : \beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$$

$$H_1 : \text{At least one } \ \beta_j \neq 0$$

by using the F test for $j = 1, ..., 4$.

After finding the test statistic:

$$F = \frac{1}{\hat{\sigma}^2}(C\hat{\vec{\beta}})^T(C(Z^TZ)^{-1}C^T)(C\hat{\vec{\beta}}) = 229.9594$$

we also obtain the corresponding critical value:

$$F^* = (r-q)F_{r-q,n-r-1}(\alpha) = 9.578594$$

Since $F > F^*$, we will reject the null hypothesis, $H_0 : \beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$, and conclude at least one $\beta_j = 0$ at $\alpha = 0.05$.

T Test

Following the result from the F Test, we are now interested in determining which of the variates in the candidate model could potentially be an insignificant variate. We will determine this through the t-test. More rigorously, we wish to test the following hypotheses:

$$H_0 : \beta_j = 0$$

$$H_1 : \beta_j \neq 0$$

for $j = 1, ..., 4$.

Testing at significance level $\alpha = 0.05$, the sampling distribution is t_{393}. Comparing the corresponding critical value to each variate's respective test statistic, we reject the null hypothesis for $j = 3$, and fail to reject the null hypothesis for $j = 1, 2, 4$. In other words, there is sufficient evidence to show that weight (z_3) was the most significant explanatory variable in our original candidate model.

Furthermore, one can construct simultaneous confidence intervals for each variate using the confidence region and Bonferroni correction.

95% Simultaneous Confidence Intervals Based on Confidence Region

	Lower Bound	Upper Bound
β_0	34.20636	47.56124
β_1	-0.03245226	0.01119413
β_2	-0.02284596	0.03240075
β_3	-0.008631879	-0.003649114
β_4	0.1543072	0.4987402

95% Simultaneous Confidence Intervals Based on Bonferroni Correction

	Lower Bound	Upper Bound
β_0	35.71574	46.05186
β_1	-0.0275193	0.006261169
β_2	-0.01660192	0.02615671
β_3	-0.008068722	-0.004212271
β_4	-0.08049915	0.4249321

95% Prediction Interval for New Response

It is worth mentioning that this data set was used in the 1983 American Statistical Association Exposition. Therefore, we would expect this multiple linear regression model to predict more accurately for cars from that era. However, what if we were interested in observing the model's performance on cars from this era? I researched the specs on one of my childhood favorites, the 2019 Mustang GT. Let $\vec{z_0}$ be a vector containing the data on the 2019 Mustang GT.

$$\vec{z_0} = \begin{bmatrix} 1 \\ 302 \\ 460 \\ 3705 \\ 4 \end{bmatrix}$$

After constructing a 95% prediction interval for Y_0, the result is as follows:

	Lower Bound	Upper Bound
Y_0	6.710095	28.90941

In fact, the miles per gallon for the 2019 Mustang GT is approximately 19.5 mpg. Since the actual mpg is within the prediction interval, the model performed well on this particular car.

Conclusion

After conducting this analysis we have discovered several insights. First, it is apparent that the model does not need all of the variates to be an efficient model. Although we discovered that there is a relatively strong, positive linear relationship between mpg and the variates, we soon saw that we need not include all of the variates (as seen through the F test and T test). In fact, it was observed that the only statistically significant explanatory variable was weight.

One of the key details when building this model was that the cars sampled were from the 1980s, over 40 years ago. Thus, we were interested in seeing the effectiveness of the model on a car from this era. In particular, I was interested in observing the model's accuracy in predicting the mpg for a newer, higher-end car, the 2019 Mustang GT. After constructing the corresponding prediction interval, it appears that the actual mpg of the Mustang GT was indeed within the interval. From this, we can conclude that cars, even from over four decades, have not changed too drastically (in terms of these four variates).

Part 2: Two Sample Test and LDA

Introduction

In this section, we will analyze the Kaggle data set *Student's Performance in Exams*. In particular, we will compare the average test scores (math, reading, writing) for students who took a test preparation course and those who did not via two-sample test. Then, we will utilize Linear Discriminant Analysis (LDA) to try and classify each type of student based on their average scores.

Summary

The goal of this part of the project is to eventually determine if there is a difference in average test scores for students who took a test course score and those who did not. We begin with summary statistics for each group.

Test Prep Students

	Math	Reading	Writing
Min	23	37	36
Q1	60	65	66
Median	69	75	76
Mean	69.7	73.89	74.42
Q3	79	84	83
Max	100	100	100

Non-test Prep Students

	Math	Reading	Writing
Min	0	17	10
Q1	54	57	54
Median	64	67	65
Mean	64.08	66.53	64.5
Q3	74.75	76	74
Max	100	100	100

At first glance of the summary statistics, it appears that the students that took a test prep course have higher scores, on average.

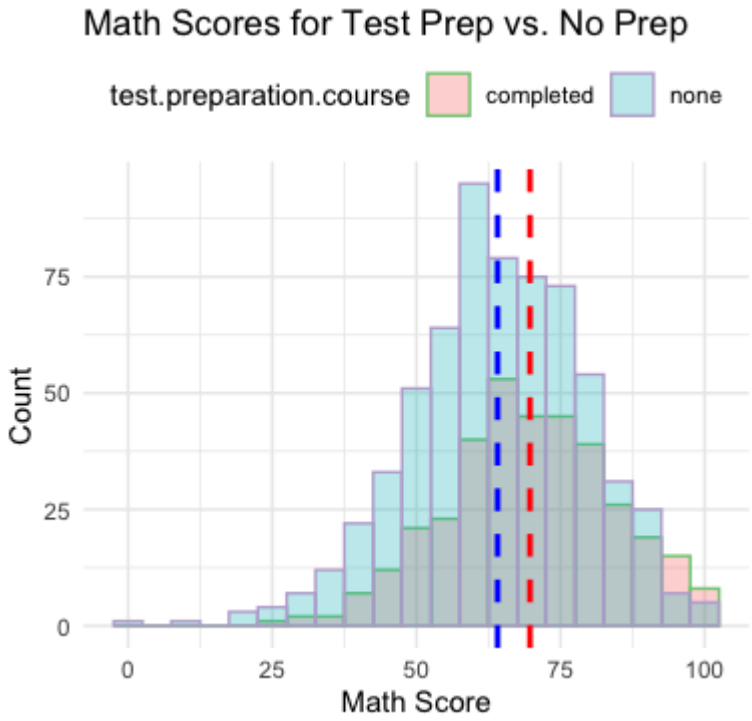
Plotted on pages 11 and 12 are the distributions of scores for each group. The histograms marked in red represent the distribution of scores for students who completed a test preparation course. Similarly, the blue histograms represent the distributions of those who did not complete one. Furthermore, the red and blue dashed lines represent the average score for students who completed and students who did not complete respectively.

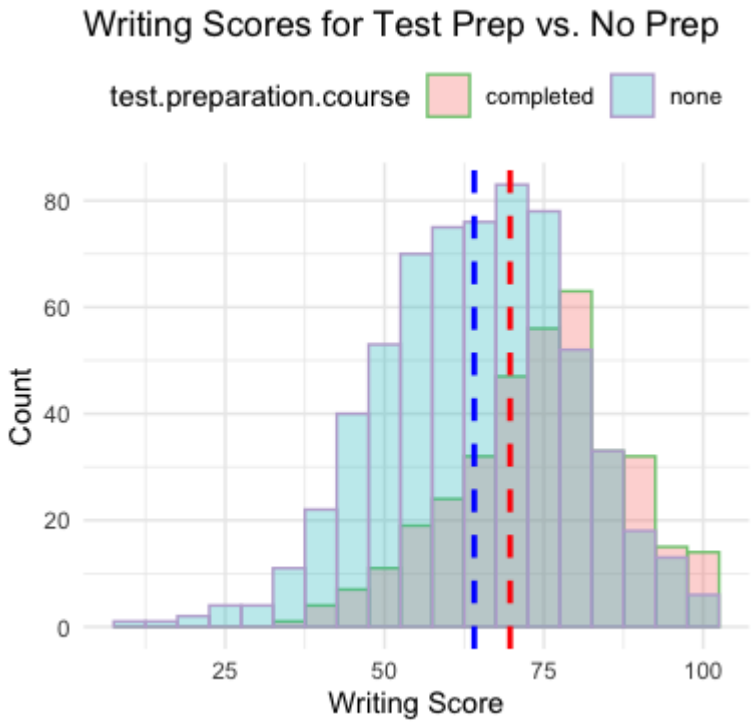
Similar to when observing the summary statistics, notice that the average test score for those who completed a test preparation course is higher than the other group in every subject (as displayed with the red and blue dashed lines).

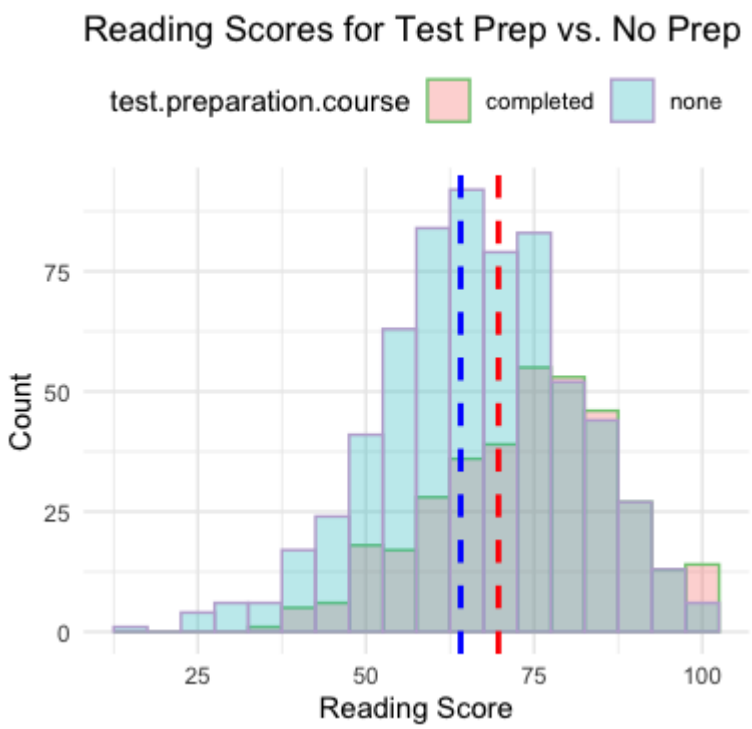
Analysis

Although we have observed through summary statistics and visualizations

Final Project







that there appears to be a difference between both groups, we wish to prove this rigorously via two-sample test. In particular, we wish to test the following hypotheses:

$$H_0 : \vec{\mu_1} - \vec{\mu_2} = \vec{0}$$

$$H_1 : \vec{\mu_1} - \vec{\mu_2} \neq \vec{0}$$

where $\vec{\mu_1}$ is average test scores for prep course students and $\vec{\mu_2}$ is average test scores for students who did not take a prep course.

After calculating Hotelling's T^2 test statistic, and the corresponding sampling distribution with $\alpha = 0.05$, we obtain the following result:

$$T^2 = 161.8241 > \frac{(998)3}{996} F_{3,996}(0.05) = 7.8573$$

Therefore, we will reject the null hypothesis at $\alpha = 0.05$ and conclude that there is indeed a difference between the average test scores of both groups. This is consistent with the initial inspection of the summary statistics and graphs.

We can further verify this result by obtaining simultaneous confidence for the mean differences in each subject.

Simultaneous Confidence Intervals Based on Confidence Region

	Lower Bound	Upper Bound
Math	2.857	8.378
Reading	4.739	9.980
Writing	7.244	12.584

Simultaneous Confidence Intervals Based on Bonferroni Correction

	Lower Bound	Upper Bound
Math	3.256	7.979
Reading	5.118	9.602
Writing	7.630	12.198

Notice that both component-wise confidence intervals do not contain 0, so we can conclude that there are significant differences in test scores between the two groups.

Linear Discriminant Analysis

After determining that there is a significant difference between the two groups,

we are now interested in trying to classify each class of student based on their average test score via Linear Discriminant Analysis. In particular, we will utilize Fisher's rule for classification. Let

$$\bar{\vec{x_1}} = \begin{bmatrix} 69.69553 \\ 73.89385 \\ 74.41899 \end{bmatrix}, \bar{\vec{x_2}} = \begin{bmatrix} 64.07788 \\ 66.53427 \\ 64.50467 \end{bmatrix}, \mathbf{S}_{pooled}^{-1} = \begin{bmatrix} 157.2008 & 120.3783 & 119.5792 \\ 120.3783 & 141.3725 & 136.0852 \\ 119.5792 & 136.0852 & 144.4775 \end{bmatrix}$$

where $\bar{\vec{x_1}}$ and $\bar{\vec{x_2}}$ represent the sample mean vectors for the test scores of the test prep and non-test prep students respectively, and $\mathbf{S}_{pooled}^{-1}$ is the pooled (inverse) sample covariance matrix.

The method in which a student ($\vec{x_0}$) is classified is as follows:

If:

$$\vec{w}^T \vec{x_0} \geq \frac{1}{2} \vec{w}^T (\bar{\vec{x_1}} + \bar{\vec{x_2}})$$

we will **classify the student as one who took a prep course**.

Otherwise:

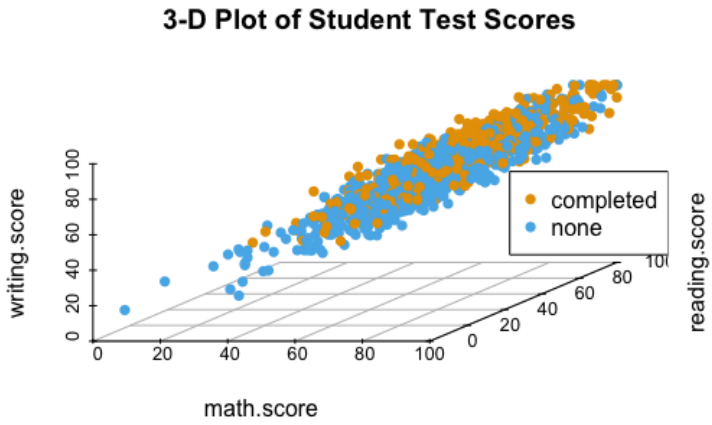
we will classify the student as one who did not take a prep course.

Using this classification method on each of the students in the data set, we observed that 657 of the 1000 students were correctly classified. In other words, there was a 34.3% training error.

Conclusion

There are two main takeaways from this analysis. We began by trying to compare the mean test scores for each class of students. After performing a two-sample test, we concluded that there was a significant difference between the two classes of students. *In other words, we can assume that the test preparation course lead to an improvement in test scores of students, on average.* This result could further be seen by the simultaneous confidence intervals as well as the plots showing that the average test scores were consistently higher for the first class in each subject.

After determining that there was an average mean difference in test scores, the natural next step was to use each group's average test scores for classification. However, after using Fisher's Rule to classify each of the students, we obtained a relatively high training error rate of approximately 34%.



Plotted above is a 3-D scatter plot of each student's subject score. Notice here that there does not appear to be any obvious cluster or grouping between

the two classes. In fact, both groups appear to be very mixed. With Fisher's Rule, we utilize the Mahalanobis Distance to determine a hyperplane that will best separate the data. Visually, we can see that it would be difficult to find a single hyperplane that could efficiently separate both groups. Therefore, this will result in a fairly high misclassification rate, as we have observed.

Furthermore, it is worth mentioning that the sample mean vectors for each score are somewhat close. That is, the average test score in each subject for both groups are not significantly far from each other. Since one of the main foundations for Fisher's equation is the sample mean vector, this "closeness" in each group's sample mean vector makes it difficult for the model to accurately classify. In other words, if the sample mean vectors for each group had a wider difference, then the model should have more correct classifications, on average.

Part 3: Principal Component Analysis

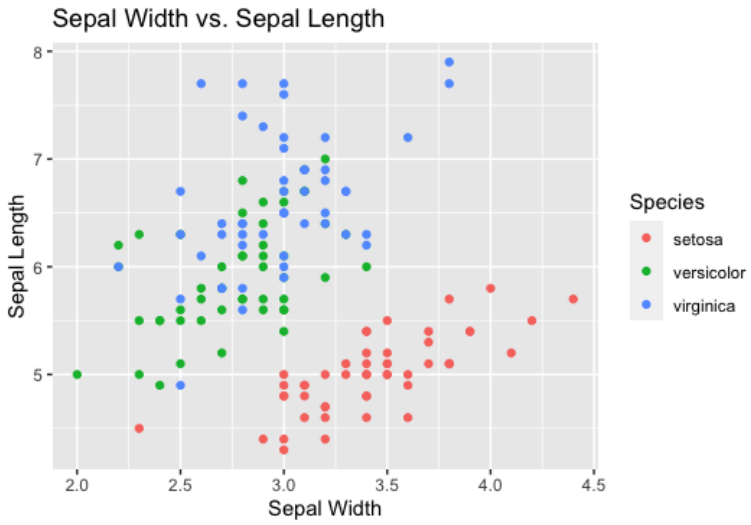
Introduction

In this section, we will analyze the *Iris* data set also from the UC Irvine Machine Learning Repository. *Iris* consists of 4 variates that describe 150 different iris flowers. More specifically, the four variates are: sepal width/length and petal width/length. We will utilize a dimension reduction technique, principal component analysis (PCA), to determine if the flowers can be described by a fewer amount of variates.

Final Project

Summary

The goal of PCA, in general is to reduce the number of explanatory variables to a smaller number of what are called principal components.



The plot above is a scatter plot of two explanatory variables from the data, sepal width and sepal length. Notice that each species of flower is generally clustered together. However, notice that with the species Setosa and Versicolor, there is a bit of mixture between the two classes. One of the goals of PCA is to be able to capture relatively similar patterns while also decreasing the dimensionality of the data. In other words, PCA seeks to capture a sizable portion of the variation while simplifying the model.

Analysis

To calculate each principal component, we calculate:

$$Y_k = \vec{v}_k^T \vec{X}$$

for k = 1,...,4, where $\vec{v}_k^T$ is the eigenvector corresponding to the k_{th} largest eigenvalue λ_k. Note also that $Var(Y_k) = \lambda_k$.

The exact values of each principal component is not particularly important, but can be seen in the through the Code Appendix. In determining the importance of the principal components, we observe the proportion of variance each principal component contributes to the overall variance. Since $Var(Y_k) = \lambda_k$, the proportion of total variance the first k principal components is defined as:

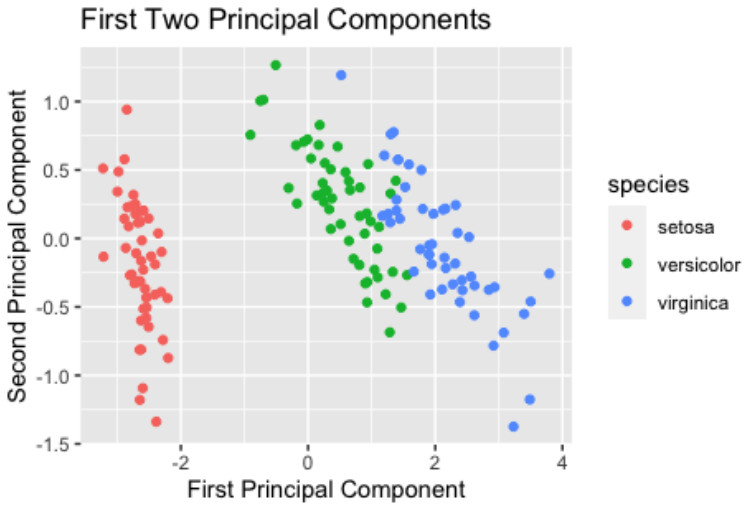
$$\frac{\lambda_1 + ... + \lambda_k}{\lambda_1 + ... + \lambda_p}$$

In particular,

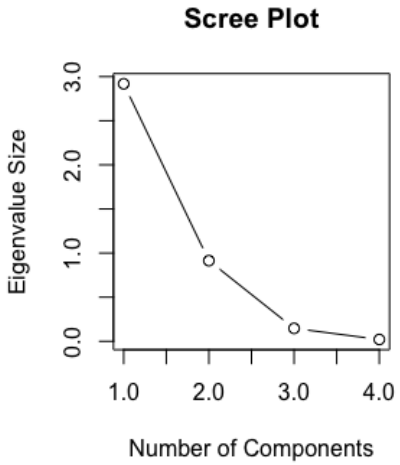
	PC1	PC2	PC 3	PC4
Standard Deviation	1.7083611	0.9560494	0.38308860	0.143926497
Proportion of Variance	0.7296245	0.2285076	0.03668922	0.005178709
Cumulative Proportion	0.7296245	0.9581321	0.99482129	1.000000000

It is worth noting that the cumulative proportion for the first two principal components is 0.9581321. In other words, approximately 95.8% of the data can be explained by the first 2 principal components. Thus we could potentially reduce the number of variates from the 4 original variates to just 2 principal components *without losing too much information.*

Plotted on the next page is a scatter plot of the first 2 principal components. Notice that these principal components have a similar effectiveness in grouping the species when comparing to the first scatter plot. This shows that using the first two principal components could be a viable option.



We can further verify the importance of the first two principal components through the scree plot shown below. A scree plot is used to determine the number of principal components should is appropriate. The rule of thumb with scree plots is to use the number of principal components to the left of its "elbow." Since the first two principal components lie to the left of the "elbow," this is the amount that should be used.



Conclusion

Now that we have determined the importance of the first two principal components, we will try to interpret them through their respective loadings. Note that loadings for the second principal component for petal length and height are empty because the result was close to zero.

	PC 1	PC 2
Sepal Length	0.521	0.377
Sepal Width	-0.269	0.923
Petal Length	0.580	
Petal Height	0.565	

With this, we can equate each principal component as a linear combination of its loadings and the original variates.

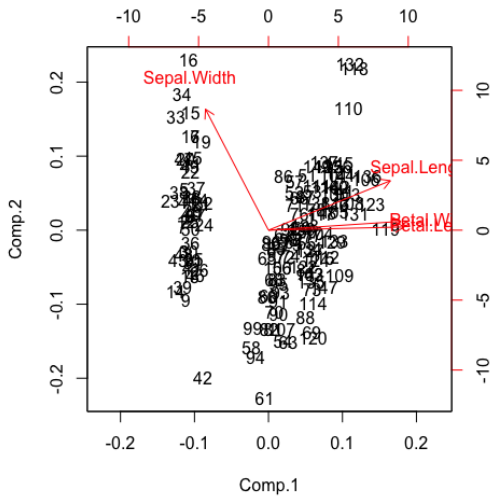
$$Y_1 = 0.521z_1 - 0.269z_2 + 0.580z_3 + 0.565z_4$$

$$Y_2 = 0.377z_1 + 0.923z_2$$

The first principal component is showing generally similar, positive loadings, except for sepal width, which is negative. The second principal component is showing that the two main contributors are sepal width and sepal length, with sepal length having a staggeringly high contribution.

We can further observe the relationship between the variates and the principal

components through the bi-plot shown below.



Paying close attention to the direction of each of the vectors, it can be seen that sepal width is mainly attributed with the second principal component. On the other hand, it appears that the other three features explain the first principal component more. Furthermore, the small angle between petal width and petal length indicate that the two features have a high positive correlation. In contrast, it appears that sepal width and sepal length are close to orthogonal to one another, showing that those features do not share much correlation.

After performing this principal component analysis, we have determined that we can indeed reduce the dimension of our data down to two principal components while still capturing approximately 95% of the data.

Code Appendix

```
# PART 1: MULTIPLE LINEAR REGRESSION

setwd('~/Downloads')
data = read.delim('auto-mpg.data', header = FALSE, sep = "")
data = data[ , c(1,3:6)]
colnames(data) = c('Y', 'z1', 'z2', 'z3', 'z4')
data$z2 = as.numeric(data$z2)

Y = data$Y
n = length(Y)
Z = cbind(rep(1,n), data[ ,2:5])
Z = data.matrix(Z)
r = dim(Z)[2]-1

# Summary of the Data
summary(Y)
summary(Z)

library(ggplot2)
# Histogram of MPG (Y)
p = ggplot(data = data, aes(x = Y))+
  geom_histogram(binwidth = 2, color="black", fill="white")+
```

```
  geom_vline(aes(xintercept=mean(Y)), color="blue",
  linetype="dashed", size=1)+
  labs(title="Distribution of Miles Per Gallon (mpg)", x="mpg",
  y = "Count")

# Histogram of Displacement (z1)
p = ggplot(data = data, aes(x = z1))+
  geom_histogram(binwidth = 75,color="black", fill="white")+
  geom_vline(aes(xintercept=mean(z1)), color="blue",
  linetype="dashed", size=1)+
  labs(title="Distribution of Displacement",
  x = "Displacement", y = "Count")

# Histogram of Horsepower (z2)
p = ggplot(data = data, aes(x = z2))+
  geom_histogram(binwidth = 20, color="black", fill="white")+
  geom_vline(aes(xintercept=mean(z2)), color="blue",
  linetype="dashed", size=1)+
  labs(title="Distribution of Horsepower", x = "Horsepower",
  y = "Count")

# Histogram of Weight (z3)
p = ggplot(data = data, aes(x = z3))+
  geom_histogram(binwidth = 300, color="black", fill="white")+
```

```
  geom_vline(aes(xintercept=mean(z3)), color="blue",
  linetype="dashed", size=1)+
  labs(title="Distribution of Weight", x = "Weight",
  y = "Count")

# Histogram of Acceleration (z4)
p = ggplot(data = data, aes(x = z4))+
  geom_histogram(binwidth = 2.5,color="black", fill="white")+
  geom_vline(aes(xintercept=mean(z4)), color="blue",
  linetype="dashed", size=1)+
  labs(title="Distribution of Acceleration",
  x = "Acceleration", y = "Count")

# Least Squares Estimates, Beta Hats
beta_hat <- solve(t(Z)%*%Z)%*%t(Z)%*%Y
beta_hat

# R^2 statistic
R_squared <- 1 - sum((Y - Z%*%beta_hat)^2)/sum((Y-mean(Y))^2)
R_squared

# Sigma_Hat Squared
sigma_hat_square <- sum((Y - Z%*%beta_hat)^2)/(n-r-1)
sigma_hat_square
```

```
# Cov(Beta_hat)
estimated_cov = sigma_hat_square * solve(t(Z)%*%Z)
estimated_cov


alpha <- 0.05
# 95% One-At-A-Time CI for Beta_0
j <- 0
cat('[',
    beta_hat[j+1] - qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_0 based on CONFIDENCE REGION
j <- 0
cat('[',
    beta_hat[j+1] - sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
```

```
    ']')


# 95% Simultaneous CI for Beta_0 based on BONFERRONI CORRECTION
j <- 0
cat('[',
    beta_hat[j+1] - qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% One-At-A-Time CI for Beta_1
j <- 1
cat('[',
    beta_hat[j+1] - qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_1 based on CONFIDENCE REGION
j <- 1
```

```
cat('[',
    beta_hat[j+1] - sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_1 based on BONFERRONI CORRECTION
j <- 1
cat('[',
    beta_hat[j+1] - qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% One-At-A-Time CI for Beta_2
j <- 2
cat('[',
    beta_hat[j+1] - qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
```

```
    beta_hat[j+1] + qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_2 based on CONFIDENCE REGION
j <- 2
cat('[',
    beta_hat[j+1] - sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_2 based on BONFERRONI CORRECTION
j <- 2
cat('[',
    beta_hat[j+1] - qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')
```

```
# 95% One-At-A-Time CI for Beta_3
j <- 3
cat('[',
    beta_hat[j+1] - qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/2, n-r-1)*sqrt(sigma_hat_square *
    solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_3 based on CONFIDENCE REGION
j <- 3
cat('[',
    beta_hat[j+1] - sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_3 based on BONFERRONI CORRECTION
j <- 3
cat('[',
    beta_hat[j+1] - qt(1-alpha/(2*(r+1)), n-r-1)*
```

```
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% One-At-A-Time CI for Beta_4
j <- 4
cat('[',
    beta_hat[j+1] - qt(1-alpha/2, n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/2, n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')

# 95% Simultaneous CI for Beta_4 based on CONFIDENCE REGION
j <- 4
cat('[',
    beta_hat[j+1] - sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + sqrt((r+1)*qf(1-alpha, r+1, n-r-1))*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
```

```
    ']')

# 95% Simultaneous CI for Beta_4 based on BONFERRONI CORRECTION
j <- 4
cat('[',
    beta_hat[j+1] - qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/(2*(r+1)), n-r-1)*
    sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']')
# 95% CI for new Mean Response

# mustang mpg is 20
z_0 = c(1,302,460, 3705,4)

# Prediction Interval for New Response
cat('[',
    z_0%*%beta_hat - sqrt(sigma_hat_square)*
    sqrt(1+t(z_0)%*%solve(t(Z)%*%Z)%*%z_0)*qt(1-alpha/2, n-r-1),
    ',',
    z_0%*%beta_hat + sqrt(sigma_hat_square)*
    sqrt(1+t(z_0)%*%solve(t(Z)%*%Z)%*%z_0)*qt(1-alpha/2, n-r-1),
    ']')
```

Final Project

```
# T Test for each variable

j <- 1
t_stat1 <- (beta_hat[j+1] - 0)/sqrt(sigma_hat_square *
solve(t(Z)%*%Z)[j+1,j+1])

j <- 2
t_stat2 <- (beta_hat[j+1] - 0)/sqrt(sigma_hat_square *
solve(t(Z)%*%Z)[j+1,j+1])

j <- 3
t_stat3 <- (beta_hat[j+1] - 0)/sqrt(sigma_hat_square *
solve(t(Z)%*%Z)[j+1,j+1])

j <- 4
t_stat4 <- (beta_hat[j+1] - 0)/sqrt(sigma_hat_square *
solve(t(Z)%*%Z)[j+1,j+1])



alpha <- 0.05
cval_t <- qt(1-alpha/2, n-r-1)

# F-test
```

```
# H_0: beta_1 = beta_2 = 0

C = rbind(c(0,1,0,0,0),
          c(0,0,1,0,0),
          c(0,0,0,1,0),
          c(0,0,0,0,1))

df_1 <- qr(C)$rank # df_1: rank of matrix R

f_stat <- (t(C%*%beta_hat)%*%solve(C%*%solve(t(Z)%*%Z)
%*%t(C))%*%(C%*%beta_hat)/df_1)/sigma_hat_square

cval_f <- qf(1-alpha, df_1, n-r-1)
cval_f * (df_1)

##########################################

# PART 2: TWO SAMPLE TEST, LINEAR DISCRIMINANT ANALYSIS

setwd('~/Downloads')
data = read.csv('StudentsPerformance.csv', header = TRUE, sep = '

# TWO SAMPLE TEST
```

```
df = data[data$test.preparation.course == "completed" |
data$test.preparation.course == 'none', ]
prep = df[df$test.preparation.course == "completed", 6:8]
noPrep = df[df$test.preparation.course == "none", 6:8]

# Summary
table(df$test.preparation.course)
summary(prep)
summary(noPrep)

# Plot the distribution of scores (Math, Writing, Reading)
ggplot(df, aes(x = math.score)) +
  geom_histogram()

ggplot(df, aes(x = writing.score)) +
  geom_histogram()

ggplot(df, aes(x = reading.score)) +
  geom_histogram()

# Plot the distribution of scores by group

p = ggplot(df, aes(x = math.score,
fill = test.preparation.course, color = test.preparation.course)
```

```
  geom_histogram(position = 'identity',binwidth = 5,
  alpha = 0.3)+
  geom_vline(aes(xintercept=mean(
  df$math.score[df$test.preparation.course == 'completed'])),
  color="red", linetype="dashed", size=1)+
  geom_vline(aes(xintercept=mean(
  df$math.score[df$test.preparation.course == 'none'])),
  color="blue", linetype="dashed", size=1)+
  labs(title="Math Scores for Test Prep vs. No Prep",
  x="Math Score", y = "Count")
p + scale_color_brewer(palette="Accent") +
  theme_minimal()+theme(legend.position="top")

p = ggplot(df, aes(x = writing.score,
fill = test.preparation.course, color = test.preparation.course)
  geom_histogram(position = 'identity',binwidth = 5,
  alpha = 0.3)+
  geom_vline(aes(xintercept=mean(
  df$writing.score[df$test.preparation.course == 'completed'])),
  color="red", linetype="dashed", size=1)+
  geom_vline(aes(xintercept=mean(
  df$writing.score[df$test.preparation.course == 'none'])),
  color="blue", linetype="dashed", size=1)+
  labs(title="Writing Scores for Test Prep vs. No Prep",
```

```
  x="Writing Score", y = "Count")
p + scale_color_brewer(palette="Accent") +
  theme_minimal()+theme(legend.position="top")


p = ggplot(df, aes(x = reading.score,
fill = test.preparation.course, color = test.preparation.course)
  geom_histogram(position = 'identity',binwidth = 5,
  alpha = 0.3)+
  geom_vline(aes(xintercept=mean(
  df$reading.score[df$test.preparation.course == 'completed'])),
  color="red", linetype="dashed", size=1)+
  geom_vline(aes(xintercept=mean(
  df$reading.score[df$test.preparation.course == 'none'])),
  color="blue", linetype="dashed", size=1)+
  labs(title="Reading Scores for Test Prep vs. No Prep",
  x="Reading Score", y = "Count")
p + scale_color_brewer(palette="Accent") +
  theme_minimal()+theme(legend.position="top")


# Perform Hotelling T^2 Test
n <- c(358,642)
p <- 3


xmean1 <- colMeans(prep)
```

```
xmean2 <- colMeans(noPrep)

d <- xmean1-xmean2

S1 <- var(prep)
S2 <- var(noPrep)
Sp <- ((n[1]-1)*S1+(n[2]-1)*S2)/(sum(n)-2)

t2 <- t(d)%*%solve(sum(1/n)*Sp)%*%d

alpha<-0.05
cval <- (sum(n)-2)*p/(sum(n)-p-1)*qf(1-alpha,p,sum(n)-p-1)

# Simultaneous Confidence Intervals (Difference in Means)
wd <- sqrt(cval*diag(Sp)*sum(1/n))
Cis <- cbind(d-wd,d+wd)

wd.b <- qt(1-alpha/(2*p),n[1]+n[2]-2) *sqrt(diag(Sp)*sum(1/n))
Cis.b <- cbind(d-wd.b,d+wd.b)

# LINEAR DISCRIMINANT ANALYSIS (LDA)

library(rrcov)
library(MASS)
```

Final Project

```
# 3-D Plot of Test Scores
colors <- c("#E69F00", "#56B4E9")
colors <- colors[as.numeric(df$test.preparation.course)]
s3d = scatterplot3d(df[,6:8], pch = 16, color=colors,grid=TRUE,
box=FALSE)
legend("right", legend = levels(df$test.preparation.course),
       col =  c("#E69F00", "#56B4E9"), pch = 16)
title(main = "3-D Plot of Student Test Scores")

# Fisher's Rule:
S.u <- 357*(var(prep)+var(noPrep))/998
w <- solve(S.u)%*%(xmean1-xmean2)
w0 <- -(xmean1+xmean2)%*%w/2

# Use built in LDA function to classify based on training data
lda.obj <- lda(test.preparation.course~math.score+reading.score+
writing.score,data=df)
plda <- predict(object=lda.obj,newdata=df)

# Figure out how many students were correctly classified
predicted = data.frame(plda$class)
actual = data.frame(df$test.preparation.course)
combined = cbind.data.frame(actual,predicted)
```

Final Project

```
sum(combined$df.test.preparation.course == combined$plda.class)

##########################################

# PART 3: PRINCIPAL COMPONENT ANALYSIS

library(ggplot2)
# Remove labels from built-in Iris
iris.df = data.frame(iris)
iris.df = iris.df[,1:4]

# Plot of Sepal Width vs. Sepal Length
ggplot(iris, aes(x=Sepal.Width, y=Sepal.Length, color=Species))+
  geom_point()+
  labs(title="Sepal Width vs. Sepal Length", x = "Sepal Width",
  y = "Sepal Length")

# Obtain principal components
pca_results = data.frame(prcomp(iris.df)$x)
pca_results$species = iris$Species

# Plot the first 2 principal components
ggplot(pca_results, aes(x=PC1, y=PC2, color=species))+
  geom_point()+
```

```
  labs(title="First Two Principal Components",
  x = "First Principal Component",
  y = "Second Principal Component")

# Obtain proportion of variance/loadings
iris.pc <- princomp(iris.df, cor=TRUE)
summary(iris.pc, loadings = TRUE)

# Scree Plot
plot(1:(length(iris.pc$sdev)), (iris.pc$sdev)^2, type='b',
      main="Scree Plot", xlab="Number of Components",
      ylab="Eigenvalue Size")

par(pty="s")
plot(iris.pc$scores[,1], iris.pc$scores[,2],
      xlab="PC 1", ylab="PC 2", type ='n', lwd=2)
text(iris.pc$scores[,1], iris.pc$scores[,2],
labels = colnames(iris.df), cex=0.7, lwd=2)

biplot(iris.pc)
```