# Introduction to Multivariate Data Analysis

Final Project Report

José Antonio Álvarez Ocete - 917933752
jocete@ucdavis.edu

June 9, 2020

## Contents

# 1 Multiple Linear Regression

## 1.1 Introduction

In this first section, we will conduct a multiple linear regression following question 1 of the 5th homework assignment. We will estimate the regression coefficients (betas) following different several methods seen during the lectures and we will provide an estimation for a new response.

## 1.2 Summary

For this first analysis, I've selected the Battery Failure dataset. In this example, we want to predict the cycles of life of a certain battery before it fails. We are provided the following variables: Charge rate (amps), discharge rate (amps), depth of discharge (% of rated ampere-hours), temperature (ºC), and end of charge voltage (volts). These are the first three rows of data:

| Charge rate (amps) | Discharge Rate (amps) | Depth of Discharge (% of rated ampere-hours) | Temperature (ºC) | End of Charge Voltage (volts) | Cicles to failure |
|---|---|---|---|---|---|
| 0.375 | 3.13 | 60.0 | 40 | 2.00 | 101 |
| 1.000 | 3.13 | 76.8 | 30 | 1.99 | 141 |
| 1.000 | 3.13 | 60.0 | 20 | 2.00 | 96 |

## 1.3 Analysis

### (1) Find the least square estimate beta hat

We obtain the least square estimate beta hat following out notes:

$$\hat{\vec{\beta}} = (Z^T \cdot Z)^{-1} \cdot Z^T \cdot \vec{Y}$$

Obtaining:

$$\hat{\vec{\beta}} = \begin{pmatrix} -2937.7571 \\ -33.7934 \\ -0.1798 \\ -1.7397 \\ 7.0627 \\ 1529.2897 \end{pmatrix}$$

### (2) Find the $R^2$ statistic

We use:

$$R^2 = \frac{||\hat{\vec{Y}} - \bar{Y} \cdot \vec{1_n}||^2}{||\vec{Y} - \bar{Y} \cdot \vec{1_n}||^2}$$

Obtaining $R^2 = 0.4799$.

### (3) Find sigma_hat_square and estimated Cov(beta square)

We use:

$$\hat{\sigma}^2 = \frac{1}{n-r-1} ||\hat{\vec{\epsilon}}||^2$$

and

$$\hat{Cov}(\hat{\vec{\beta}}) = \hat{\sigma}^2 (Z^T Z)^{-1}$$

We obtain the following:

$$\hat{\sigma}^2 = 7138.186$$

$$\hat{Cov}(\hat{\vec{\beta}}) = \begin{pmatrix} 1.633e+07 & -2933.74 & 2980.4460 & -34.78143 & -991.73637 & -8.160e+06 \\ -2.934e+03 & 1880.55 & 18.5503 & 17.34897 & 10.28445 & -1.764e+02 \\ 2.980e+03 & 18.55 & 193.4117 & -3.23257 & 0.34449 & -1.696e+03 \\ -3.478e+01 & 17.35 & -3.2326 & 1.79944 & -0.08092 & -4.242e+01 \\ -9.917e+02 & 10.28 & 0.3445 & -0.08092 & 3.89193 & 4.549e+02 \\ -8.160e+06 & -176.39 & -1695.7845 & -42.42251 & 454.86652 & 4.081e+06 \end{pmatrix}$$

**(4) 95% confidence interval for each $\beta_j$**

We use one at a time confidence intervals for the betas:

$$\beta_j \in [\hat{\beta}_j \pm \hat{\sigma} \cdot \sqrt{\omega_{jj}} \cdot t_{n-r-1}(\frac{\alpha}{2})]$$

Obtaining:

$$\begin{aligned} \beta_0 &\in [-11604, 5729] \\ \beta_1 &\in [-126.8, 59.22] \\ \beta_2 &\in [-30.01, 29.65] \\ \beta_3 &\in [-4.617, 1.137] \\ \beta_4 &\in [2.831, 11.29] \\ \beta_5 &\in [-2804, 5862] \end{aligned}$$

**(5) 95% simultaneous confidence intervals for all betas based on the confidence region**

Using the formula from the notes:

$$\beta_j \in [\hat{\beta}_j \pm \hat{\sigma} \cdot \sqrt{\omega_{jj}} \cdot \sqrt{(r+1) \cdot F_{r+1,n-r-1}(\alpha)}]$$

We obtain:

$$\begin{aligned} \beta_0 &\in [-19640, 13764] \\ \beta_1 &\in [-213, 145.5] \\ \beta_2 &\in [-57.67, 57.31] \\ \beta_3 &\in [-7.285, 3.805] \\ \beta_4 &\in [-1.092, 15.22] \\ \beta_5 &\in [-6822, 9880] \end{aligned}$$

**(6) 95% simultaneous confidence intervals for all betas based on the Bonferroni correction**

We compute a final set of intervals for the ebtas using the Bonferroni correction:

$$\beta_j \in [\hat{\beta}_j \pm \hat{\sigma} \cdot \sqrt{\omega_{jj}} \cdot t_{n-r-1}(\frac{\alpha}{2(r+1)})]$$

We obtain:

$$\begin{aligned}
\beta_0 &\in [-15338, 9462] \\
\beta_1 &\in [-166.9, 99.29] \\
\beta_2 &\in [-42.86, 42.5] \\
\beta_3 &\in [-5.856, 2.377] \\
\beta_4 &\in [1.009, 13.12] \\
\beta_5 &\in [-4670, 7729]
\end{aligned}$$

**(7) Test $H_0 : \beta_1 = \beta_2 = 0$ at significance level $\alpha = 0.05$**

Using this matrix for the linear transformation:

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We can compute the F-test statistic:

$$\vec{\beta}_{(2)}^T \Omega_{22}^{-1} \vec{\beta}_{(2)} = 108296$$

And compare it to:

$$(r-q) \cdot \hat{\sigma}^2 \cdot F_{r-q,n-r-1}(\alpha) = 133445$$

Since $108296 < 133445$, we don't have sufficient evidence to assure that $\beta_1 = \beta_2 = 0$.

**(8) 95% confidence interval for the mean response given $\mathbb{E}(Y_0) = \beta_0 + \sum_{i=1}^5 \beta_i \cdot \bar{z}_i$, where $\bar{z}_i$ is the sample mean of $z_{i,j}$ for $i \in \{1, ..., n\}$**

First, compute $\vec{z_0}$:

$$\vec{z_0} = \begin{pmatrix} 1.000 \\ 1.031 \\ 3.034 \\ 62.840 \\ 19.500 \\ 1.999 \end{pmatrix}$$

And now compute the confidence intervals for it's associated value using the formula in the class notes:

$$\vec{z_0}^T \vec{\beta} \in [\vec{z_0}^T \hat{\vec{\beta}} \pm \hat{\sigma} \cdot t_{n-r-1}(\frac{\alpha}{2}) \sqrt{\vec{z_0}^T (Z^T Z)^{-1} \vec{z_0}}]$$

Obtaining the following interval: $\vec{z_0}^T \vec{\beta} \in [71.78, 152.8]$

**(9) 95% confidence interval for a new response $Y_0$ given $\vec{z_0}$**

Using a similiar formula:

$$\vec{z_0}^T \vec{\beta} \in [\vec{z_0}^T \hat{\vec{\beta}} \pm \hat{\sigma} \cdot t_{n-r-1}(\frac{\alpha}{2}) \sqrt{1 + \vec{z_0}^T (Z^T Z)^{-1} \vec{z_0}}]$$

And using that $Y_0 = \vec{z_0}^T \vec{\beta} + \epsilon_0$ we obtain:

$$Y_0 \in [-73.38, 298]$$

A substantialy bigger interval than the over in **(8)**, which makes sense since we are including the error now.

## 2 Principal Component Analysis

### 2.1 Introduction

For the second analysis we will conduct Principal Component Analysis on our dataset. Ourmain objectives will be to analyze the explained variance against the numbers of principal components in order to reduce the dimensionality of our dataset, as well as study how standarizing the data would affect these parameters.

### 2.2 Summary

In order to achieve those objetives we need to select a dataset relatively hihg dimensionality, at least compare to the rest of the datasets in the book, that will also yield different results upon standarizing the data. In order to fullfill this criteria I selected the Breakfast Cereal dataset.

We this dataset we aim to distinguish cereals from three different manufacturers: General Mills (G), Kellogg (K), and Quaker (Q); by looking at 8 variates: Calories, Protein, Fat, Sodium, Fiber, Carbohydrates, Sugar, and Potassium. These are the first 3 rows of each manufacturer:

| Manufacturer | Calories | Protein | Fat | Sodium | Fiber | Carbohydrates | Sugar | Potassium |
|---|---|---|---|---|---|---|---|---|
| G | 110 | 2 | 2 | 180 | 1.5 | 10.5 | 10 | 1 |
| G | 110 | 6 | 2 | 290 | 2.0 | 17.0 | 1 | 1 |
| G | 110 | 1 | 1 | 180 | 0.0 | 12.0 | 13 | 55 |
| K | 70 | 4 | 1 | 260 | 9.0 | 7.0 | 5 | 320 |
| K | 110 | 2 | 0 | 125 | 1.0 | 11.0 | 14 | 30 |
| K | 100 | 2 | 0 | 290 | 1.0 | 21.0 | 2 | 35 |
| Q | 120 | 1 | 2 | 220 | 0.0 | 12.0 | 12 | 35 |
| Q | 120 | 1 | 2 | 220 | 1.0 | 12.0 | 11 | 45 |
| Q | 100 | 4 | 2 | 150 | 2.0 | 12.0 | 6 | 95 |

Although our data is quite unbalanced class-wise, this doesn't affect our PCA analysis.

### 2.3 Analysis

We start by computing basic stats of our data:

$$\bar{\vec{x}} = \begin{pmatrix} 107.9070 \\ 2.4651 \\ 0.9767 \\ 180.4651 \\ 1.7140 \\ 10.2093 \\ 7.6047 \\ 84.4186 \end{pmatrix}$$

$$S = \begin{pmatrix} 359.80 & 0.76 & 5.90 & 505.76 & -0.66 & 11.40 & 49.87 & 177 \\ 0.76 & 1.49 & 0.20 & 9.06 & 1.13 & 0.47 & -2.22 & 40 \\ 5.90 & 0.20 & 0.64 & 0.61 & 0.24 & -1.16 & 0.68 & 17 \\ 505.76 & 9.06 & 0.61 & 6274.78 & 6.18 & 182.28 & -17.55 & 599 \\ -0.66 & 1.13 & 0.24 & 6.18 & 3.24 & 0.92 & -0.28 & 110 \\ 11.40 & 0.47 & -1.16 & 182.28 & 0.92 & 20.88 & -5.30 & 20 \\ 49.87 & -2.22 & 0.68 & -17.55 & -0.28 & -5.30 & 20.58 & 24 \\ 177.33 & 40.40 & 16.53 & 599.09 & 110.48 & 19.53 & 24.41 & 4370 \end{pmatrix}$$

In order to run PCA we compute the spectral decomposition of $S$:
$S = V\Lambda V^T$
Where:

$$V = \begin{pmatrix} -0.087 & 0.008 & 0.982 & 0.104 & 0.124 & 0.020 & -0.035 & 0.000 \\ -0.003 & 0.009 & -0.005 & 0.079 & 0.197 & -0.930 & 0.276 & 0.117 \\ -0.001 & 0.004 & 0.017 & -0.042 & 0.096 & 0.184 & 0.802 & -0.558 \\ -0.957 & -0.279 & -0.079 & -0.034 & 0.004 & -0.001 & -0.001 & -0.001 \\ -0.006 & 0.025 & -0.015 & 0.035 & -0.021 & -0.255 & -0.508 & -0.821 \\ -0.028 & -0.008 & -0.016 & 0.824 & -0.555 & -0.017 & 0.106 & -0.011 \\ 0.001 & 0.007 & 0.165 & -0.547 & -0.793 & -0.187 & 0.100 & -0.009 \\ -0.277 & 0.960 & -0.032 & -0.002 & 0.000 & 0.015 & 0.008 & 0.022 \end{pmatrix}$$

$$\Lambda = \begin{pmatrix} 6499.572 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4201.334 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 321.556 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 18.435 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9.642 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.677 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.397 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.287 \end{pmatrix}$$

As we can see, our first two eigen values are way larger than the rest. By looking at the loading in $V$ we can see how these first two principal components are essetially the first and sixth variable respectively. Selecting the first two principal components from this analysis isn't much different from selecting those two variables. As we will see, they explain the variance pretty well, but that's only because the data is not standarized and the scale makes a huge different. Using the following result:

$$\sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} \sigma_i^2$$

We can compute the ratio of explained variance by the first two principal components:

$$Explained\ variance = \frac{\lambda_1 + \lambda_2}{\sum_{i=1}^{n} \lambda_i} = 0.968$$

Let's study now how our data behaves if it's standarized. we obtain the spectral decomposition of the correlation matrix, which is the covariance matrix of the standarized data, and obtain the following:

$$V_z = \begin{pmatrix} -0.161 & -0.540 & -0.370 & 0.13 & 0.45 & -0.040 & 0.562 & -0.069 \\ -0.454 & 0.233 & 0.093 & 0.41 & 0.57 & 0.323 & -0.375 & 0.013 \\ -0.261 & -0.409 & 0.182 & 0.58 & -0.37 & -0.453 & -0.221 & -0.071 \\ -0.167 & 0.012 & -0.640 & 0.18 & -0.50 & 0.518 & -0.063 & -0.053 \\ -0.561 & 0.096 & 0.145 & -0.38 & -0.12 & -0.041 & 0.123 & -0.694 \\ -0.114 & 0.284 & -0.615 & -0.15 & 0.16 & -0.624 & -0.298 & 0.043 \\ 0.032 & -0.630 & -0.040 & -0.44 & 0.12 & 0.170 & -0.602 & -0.016 \\ -0.586 & -0.034 & 0.110 & -0.29 & -0.16 & -0.015 & 0.159 & 0.710 \end{pmatrix}$$

$$\Lambda_z = \begin{pmatrix} 2.49 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.89 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.63 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.91 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.39 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 \end{pmatrix}$$

As we can see, the eigen values are now much more closer to each other, and the loadings are way more balanced.

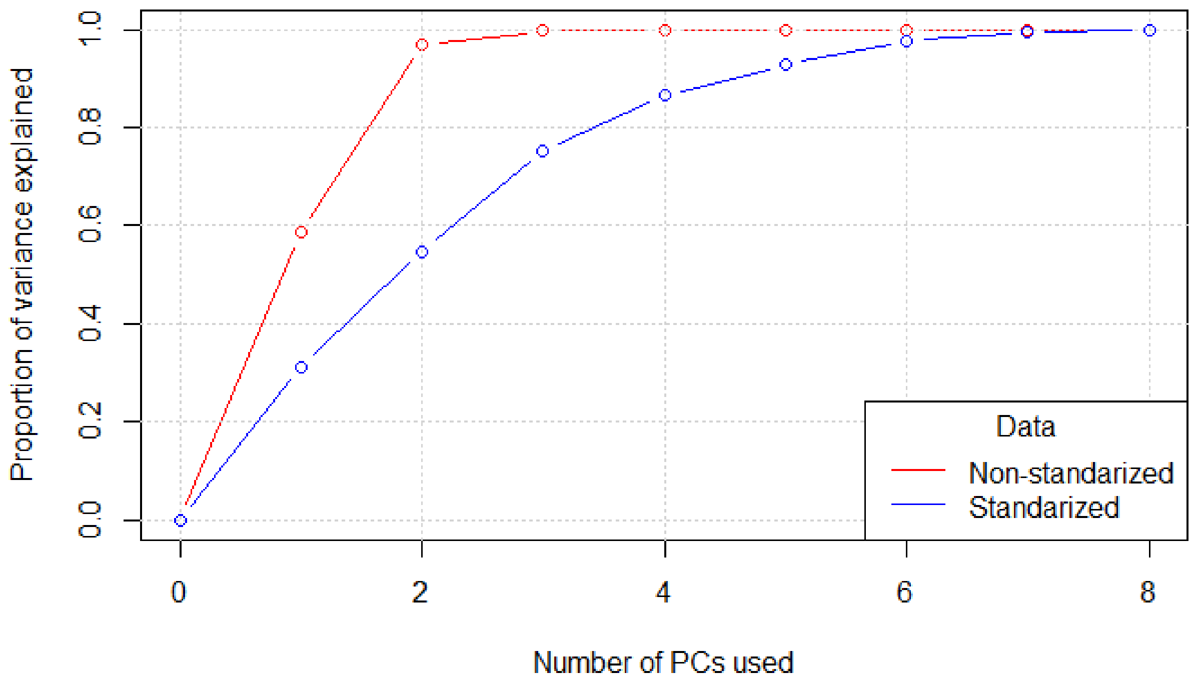


Figure 1: Portion of explained variance comparison

In 1 we see how the growth of the proportion of variance explained with the standarized data is much more progresive. As we hypothesized before, with the non-standarized analysis we were using certain variables just because their variance was bigger. Now ous principal components use information for more variates, gaining more information of the whole dataset with fewer of them.

For the rest of the anaylsis let's stick to the first two principal components. Let's transform our data using them and plot it:

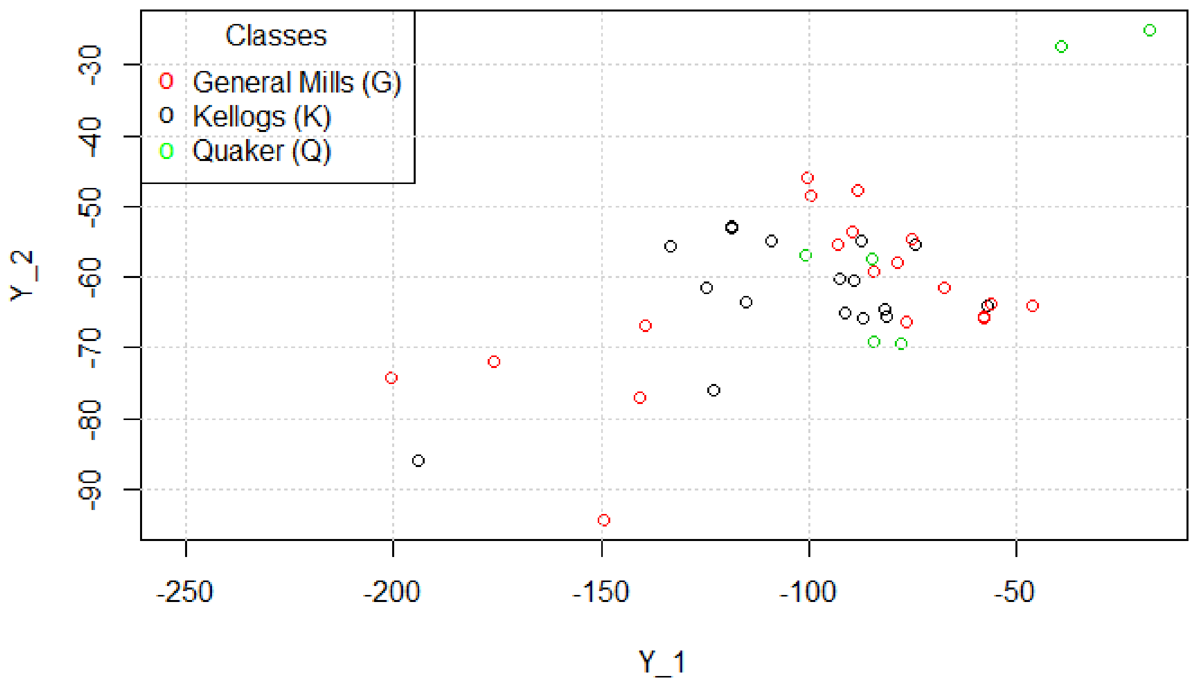


Figure 2: Portion of explained variance comparison

We can see how our classes are not clearly separated, but that's not the main point of PCA. However the points are fairly spread out. After all, we are only explaining 55% of our variance with the first two principal components. Finally, let's study how each variate contributes to these PCs. In order to do this we look at the loading of these PCs: studying only the first two columns of $V_z$ (our first two PCs), and plotting their rows we can see the variate contributions.

In 3 see each variate contribution as a colored vector. The bigger the vector, the greater the contribution, while the direction shows towards which principal component it contributes and wether it's a direct or inverse contribution.

We can see how the variates $X_4$ and $X_6$ contribute the least to the first two principal components. On one hand, $X_1$ and $X_7$ contribute mostly to the first PC, $Y_1$. On the other hand, $X_5$ and $X_8$ contribute mostly to $Y_2$. Finally, $X_2$ and $X_3$ contribute to greatly to both PCs but in very different ways.

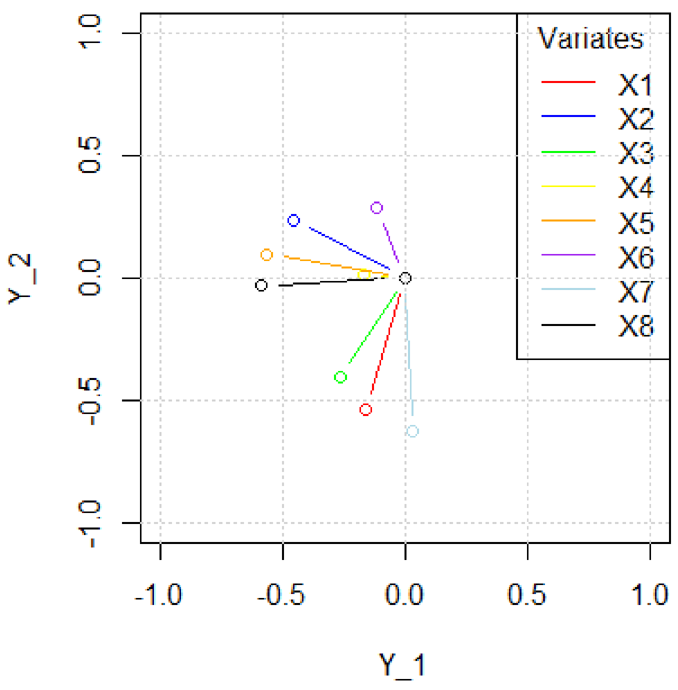


Figure 3: Portion of explained variance comparison

## 3 Two-Sample test and Linear Discriminant Analysis

### 3.1 Introduction

For this final analysis we will conduct Linear Discriminant Analysis. That is, given enough data about tagged as two different classes, classify a new reponse into one of this classes. In order to do that we first need to conduct a two sample test to make sure that our populations are actually different.

### 3.2 Summary

The dataset selection for this example was a little trickier since I wanted to display nice graphs about the data. In order to achieve this the more convinient way was to pick up a two-variate dataset with a nice visual separation between its classes. We will see why in the analysis.

The dataset selected was the Anaconda one. It has two variates: Snout vent length and weight. The class we want to predict is the snake gender: either male (M) or female (F). These are the first 5 rows of our dataset from each class:

| Snout Vent Length | Weight | Gender | Snout Vent Length | Weight | Gender |
|---|---|---|---|---|---|
| 271.0 | 18.50 | F | 176.7 | 3.00 | M |
| 477.0 | 82.50 | F | 259.5 | 9.75 | M |
| 306.3 | 23.40 | F | 258.0 | 10.07 | M |
| 365.3 | 33.50 | F | 229.8 | 7.50 | M |
| 466.0 | 69.00 | F | 233.0 | 6.25 | M |
| 440.7 | 54.00 | F | 237.5 | 9.85 | M |

Our dataset is completely balanced, having the same number of males than females.

## 3.3 Analysis

As stated in the introduction we start by conduction a two sample test with the null hypothesis $H_0 : \mu_1 = \mu_2$. We start by extracting the basic stats from the data and computing the Hotelling's $T^2$ statistic:

$$\bar{\vec{x_1}} = \begin{pmatrix} 348.28 \\ 37.26 \end{pmatrix}, \bar{\vec{x_2}} = \begin{pmatrix} 228.75 \\ 7.29 \end{pmatrix}$$

$$S_{pooled} = \begin{pmatrix} 2606.4 & 667.9 \\ 667.9 & 204.2 \end{pmatrix}$$

$$T^2 = (\bar{\vec{x_1}} - \bar{\vec{x_2}})^T ((\frac{1}{n_1} + \frac{1}{n_2}) S_{pooled})^{-1} (\bar{\vec{x_1}} - \bar{\vec{x_2}}) = 76.92$$

We compare this value to:

$$F = \frac{(n_1 + n_2 - 2)p}{n_1 + n_2 - 1 - p} \cdot F_{p, n_1 + n_2 - 1 - p}(\alpha) = 6.463$$

Since $76.92 > 6.463$, we reject $H_0$, concluding that both population are, indeed, different. We can now procede with linear discrimination analysis. For a given new response $\vec{x_0}$ we will classify it in the first population if and only if:

$$(\bar{\vec{x_1}} - \bar{\vec{x_2}})^T \cdot S_{pooled}^{-1} \cdot (\vec{x_0} - \frac{1}{2}(\bar{\vec{x_1}} - \bar{\vec{x_2}})) \geq 0$$

That is, if and only if:

$$0.05095 - 0.01986 \cdot (\vec{x_0} - \begin{pmatrix} 288.5 \\ 22.28 \end{pmatrix}) \geq 0$$

In order to meassure how good this classifier is we use two methods:

- The Apparent Error Rate (AER): Try to predict every single sample by training the model with every point in the dataset (including the one we will predict), and the use it for the prediction.
- The Expected Actual Error Rate (EAER): Using the same tachnique but exclude the point that we will predict from the train set.

In this case we obtain the same value for both measures: a $0.08929 = \frac{5}{56}$ error rate, missing 5 samples. Finally, let's plot our populations and their respective mean-centered mahalanobis-distance elipses to gain a geometry perspective of what is going on here:

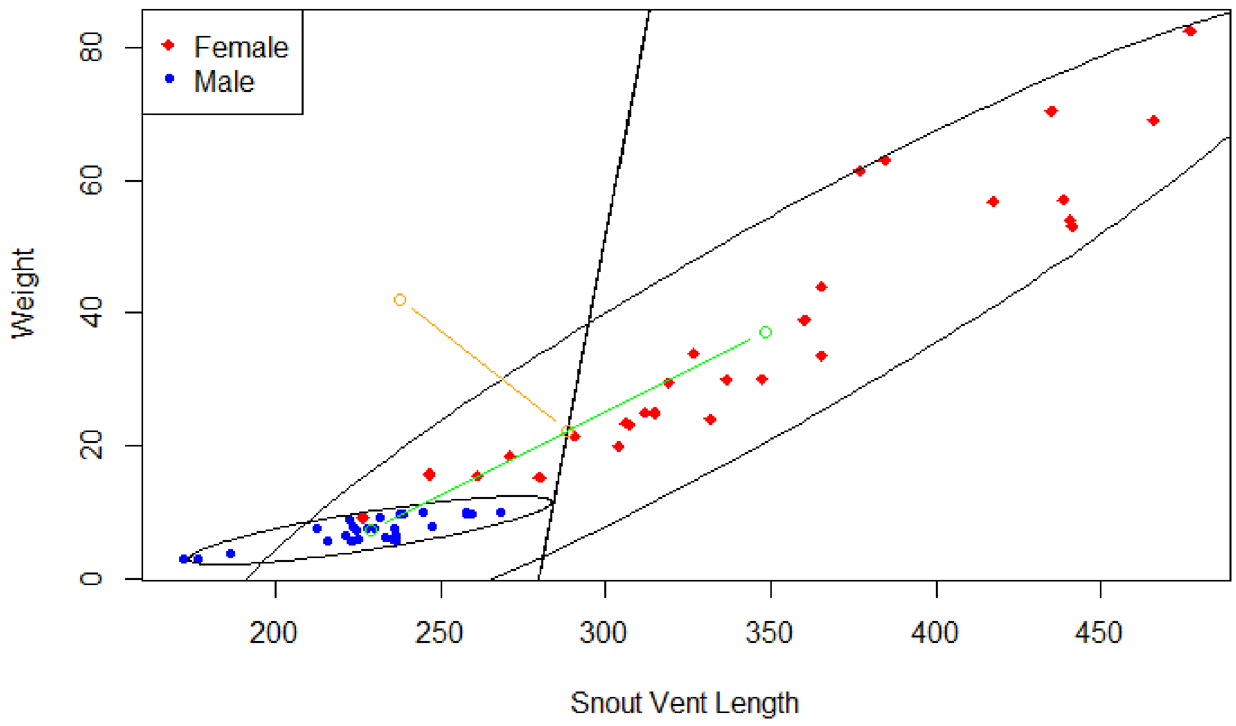


Figure 4: Anaconda dataset partition

In this figure we can see the two different classes in red (female) and blue (male). We have also plotted each population mean-centered elipse and a green line joining their centers. We can rewrite our test using the following: $\vec{x_0}$ will be classified as Female (class 1) if and only if:

$$\vec{\omega}^T \cdot \vec{\omega}_0 \geq 0$$

Where:

$$\vec{\omega} = (\bar{\vec{x_1}} - \bar{\vec{x_2}})^T \cdot S_{pooled}^{-1}$$

$$\vec{\omega}_0 = (\vec{x_0} - \frac{1}{2}(\bar{\vec{x_1}} - \bar{\vec{x_2}}))$$

What we are computing here is the cosine of the angle between $\vec{\omega}$ and $\vec{\omega}_0$. If the absolute value of the angle is less than 90º, the cosine will be positive, so we are just studying this angel. In 4, we plot $\vec{\omega}$ in orange, while $\vec{\omega}_0$ will be a vector from the middle point of the green segment to the given new responde $\vec{x_0}$.

The easiest way to see this geometrically is to plot a perpendicular line to $\vec{\omega}$. This is the black line (although it doesn't seem perpendicular, this is because of the different scale in the axis). If the point is on the right side of the line, $\vec{\omega}_0$ and $\vec{\omega}$ will create an angle smaller than 90º, classifying $\vec{x_0}$ as Female, while if the point is on the left side of the line, $\vec{\omega}_0$ and $\vec{\omega}$ will create an angle greater than 90º, classifying $\vec{x_0}$ as Male.

# Final project: Regression

Jose Antonio Alvarez Ocete

For fractions:

```r
library(MASS)
options(digits=4)
```

## R Markdown

```r
#import data
data <- read.table("data/battery.dat")

p <- 5

#set up X
X <- data.matrix(data[,1:p])
n <- length(X[,1])
Z <- cbind(rep(1,n), X)
Y <- data.matrix(data[,p+1])

names <- c('','Z1','Z2','Z3','Z4','Z5')
colnames(Z)<- names

sampleMean<-function(X, n) {
  Ones <- rep(1,n)
  return (1/n * t(X)%*%Ones)
}

sampleCovariance<-function(X, n, sample_mean) {
  Ones <- rep(1,n)
  return (1/(n-1) * t(X - Ones%*%t(sample_mean))%*%(X - Ones%*%t(sample_mean)))
}
```

(1) Find the least square estimate beta_hat

```r
# least square estimates
beta_hat <- solve(t(Z)%*%Z)%*%t(Z)%*%Y
rownames(beta_hat) <- c('beta0','beta1','beta2','beta3','beta4','beta5')
beta_hat
```

```
##             [,1]
## beta0 -2937.7571
## beta1   -33.7934
## beta2    -0.1798
## beta3    -1.7397
## beta4     7.0627
## beta5  1529.2897
```

(2) Find the R^2 statistic

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

```
## [1] 0.4799
```

(3) Find sigma_hat_square and estimated Cov(beta_square)

```
# sigma_hat_square
sigma_hat_square <- sum((Y - Z%*%beta_hat)^2)/(n-p-1)
cat('Sigma hat square:', sigma_hat_square, fill=TRUE)
```

```
## Sigma hat square: 7138
```

```
# estimated covariance of hat{beta}
covariance_hat <- sigma_hat_square * solve(t(Z)%*%Z)
cat('Covariance Hat:', fill=TRUE)
```

```
## Covariance Hat:
```

```
covariance_hat
```

```
##                       Z1         Z2        Z3         Z4         Z5
##     1.633e+07 -2933.74  2980.4460 -34.78143 -991.73637 -8.160e+06
## Z1 -2.934e+03  1880.55    18.5503  17.34897   10.28445 -1.764e+02
## Z2  2.980e+03    18.55   193.4117  -3.23257    0.34449 -1.696e+03
## Z3 -3.478e+01    17.35    -3.2326   1.79944   -0.08092 -4.242e+01
## Z4 -9.917e+02    10.28     0.3445  -0.08092    3.89193  4.549e+02
## Z5 -8.160e+06  -176.39 -1695.7845 -42.42251  454.86652  4.081e+06
```

NOT USED:

```
# t-test for single coefficient
# H_0: beta_j = 0, H_a: beta_j != 0

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

```
## [1] -0.7793
```

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

```
## [1] 2.145
```

(4) 95% confidence interval for the beta:

```
# One-at-a-time confidence interval for beta_j

for (j in 0:p) {
  cat('Beta hat', j,': [',
    beta_hat[j+1] - qt(1-alpha/2, n-p-1)*sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/2, n-p-1)*sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']', fill=TRUE)
}
```

```
## Beta hat 0 : [ -11604 , 5729 ]
## Beta hat 1 : [ -126.8 , 59.22 ]
## Beta hat 2 : [ -30.01 , 29.65 ]
## Beta hat 3 : [ -4.617 , 1.137 ]
## Beta hat 4 : [ 2.831 , 11.29 ]
## Beta hat 5 : [ -2804 , 5862 ]
```

(5) 95% simultaneous confidence intervals for all betas based on the confidence region

```
# confidence region based simultaneous confidence intervals

for (j in 0:p) {
cat('Beta hat', j,': [',
      beta_hat[j+1] - sqrt((p+1)*qf(1-alpha, p+1, n-p-1))*sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j
      ',',
      beta_hat[j+1] + sqrt((p+1)*qf(1-alpha, p+1, n-p-1))*sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j
      ']', fill=TRUE)
}
```

```
## Beta hat 0 : [ -19640 , 13764 ]
## Beta hat 1 : [ -213 , 145.5 ]
## Beta hat 2 : [ -57.67 , 57.31 ]
## Beta hat 3 : [ -7.285 , 3.805 ]
## Beta hat 4 : [ -1.092 , 15.22 ]
## Beta hat 5 : [ -6822 , 9880 ]
```

(6) 95% simultaneous confidence intervals for all betas based on the Bonferroni correction

```
# Bonferroni correction based simultaneous confidence intervals

for (j in 0:p) {
  cat('Beta hat', j,': [',
    beta_hat[j+1] - qt(1-alpha/(2*(p+1)), n-p-1)*sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ',',
    beta_hat[j+1] + qt(1-alpha/(2*(p+1)), n-p-1)*sqrt(sigma_hat_square * solve(t(Z)%*%Z)[j+1,j+1]),
    ']', fill=TRUE)
}
```

```
## Beta hat 0 : [ -15338 , 9462 ]
## Beta hat 1 : [ -166.9 , 99.29 ]
## Beta hat 2 : [ -42.86 , 42.5 ]
## Beta hat 3 : [ -5.856 , 2.377 ]
## Beta hat 4 : [ 1.009 , 13.12 ]
## Beta hat 5 : [ -4670 , 7729 ]
```

(7) Test H_0: beta_1 = beta_2 = 0 at significance level alpha = 0.05

```
# F-test
# H_0: beta_1 = beta_2 = 0

C <- cbind(rep(0,p), diag(p))

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)

cval_f <- sigma_hat_square*df_1*qf(1-alpha, 2, n-p-1)
```

```
cat (f_stat, '<', cval_f, '?', fill=TRUE)
```

```
## 108296 < 133445 ?
```

```
# (equivalent) F-test by comparing residuals

# fit the reduced model
beta_hat_reduced <- solve(t(Z[,1])%*%Z[,1])%*%t(Z[,1])%*%Y

f_stat_reduced <- ((sum((Y - Z[,1]%*%beta_hat_reduced)^2) - sum((Y - Z%*%beta_hat)^2))/2)

cat (f_stat_reduced, '<', cval_f, '?', fill=TRUE)
```

```
## 54148 < 133445 ?
```

(8) 95% confidence interval for the mean response given z_0

```
# confidence interval for z_0^T beta

X_mean <- sampleMean(X,n)
z_0 <- rbind(1, X_mean)

cat('C.I. for the mean response z_0[',
  t(z_0)%*%beta_hat - sqrt(sigma_hat_square)*sqrt(t(z_0)%*%solve(t(Z)%*%Z)%*%z_0)*qt(1-alpha/2, n-p-1),
  ',',
  t(z_0)%*%beta_hat + sqrt(sigma_hat_square)*sqrt(t(z_0)%*%solve(t(Z)%*%Z)%*%z_0)*qt(1-alpha/2, n-p-1),
  ']')
```

```
## C.I. for the mean response z_0[ 71.78 , 152.8 ]
```

(9) 95% confidence interval for a new response given z_0

```
# prediction interval for Y_0 = z_0^T beta + epsilon_0

cat('Prediction interval for Y_0 = z_0^T beta + epsilon_0: [',
  t(z_0)%*%beta_hat - sqrt(sigma_hat_square)*sqrt(1+t(z_0)%*%solve(t(Z)%*%Z)%*%z_0)*qt(1-alpha/2, n-p-1
  ',',
  t(z_0)%*%beta_hat + sqrt(sigma_hat_square)*sqrt(1+t(z_0)%*%solve(t(Z)%*%Z)%*%z_0)*qt(1-alpha/2, n-p-1
  ']')
```

```
## Prediction interval for Y_0 = z_0^T beta + epsilon_0: [ -73.38 , 298 ]
```

# Final project: PCA

Jose Antonio Alvarez Ocete

For fractions:

```
library(MASS)
options(digits=2)
```

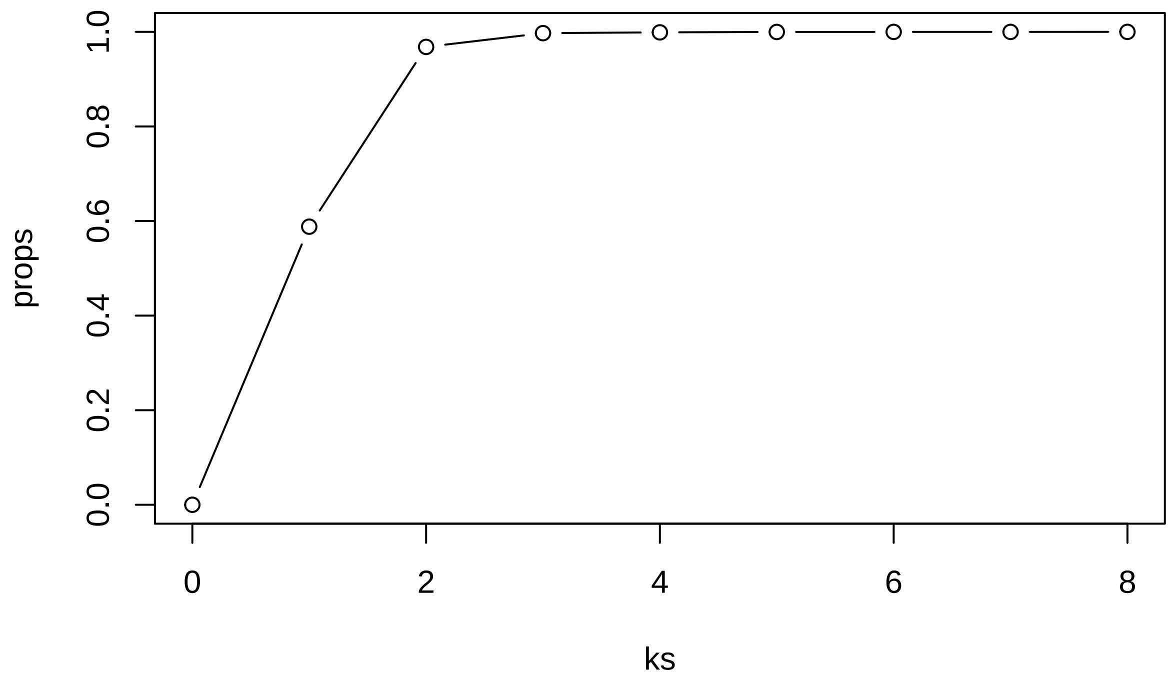
```
#import data, table T5-8.dat, see example 5.8
data <- read.table("data/cereal.dat")

r <- 8

#set up X
X <- data.matrix(data[,3:10])
colnames(X) <- c('Calories','Protein','Fat','Sodium','Fiber','Carbohydrates','Sugar','Potassium')
classes <- data.matrix(data[,11])
n <- length(X[,1])
```

(1.b) Determine the proportion of total sample variance due to the first sample principale component.

```
plotProportions<-function(eigen_values, print=FALSE) {
  ks <- c(0)
  props <- c(0)
  RR <- sum(eigen_values)

  for (k in 1:(r)) {
    ks <- c(ks, k)
    LL_r <- sum(eigen_values[1:k])
    props <- c(props, LL_r/RR)

    if (print)
      cat('k =',k,':', fractions(LL_r/RR), '=', LL_r, '/', RR, fill=TRUE)
  }
  plot(ks, props, type="b",xlim=c(0,r))
}
```

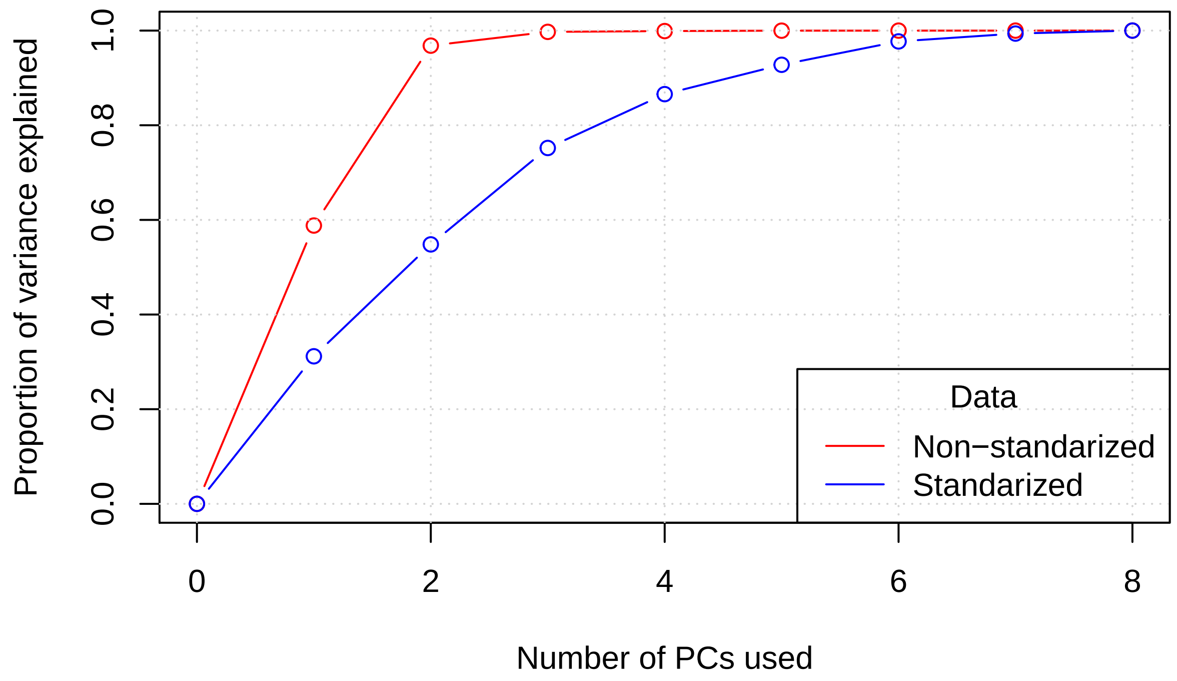
```
# Compute sample mean and S
Ones <- rep(1,n)
x_sample_mean <- 1/n * t(X)%*%Ones
S <- 1/(n-1) * t(X - Ones%*%t(x_sample_mean))%*%(X - Ones%*%t(x_sample_mean))

# eigens
ev <- eigen(S)
eigen_values <- ev$values
V <- ev$vectors

plotProportions(eigen_values, print=TRUE)
```

```
## k = 1 : 0.59 = 6500 / 11052
## k = 2 : 0.97 = 10701 / 11052
## k = 3 : 1 = 11022 / 11052
## k = 4 : 1 = 11041 / 11052
## k = 5 : 1 = 11051 / 11052
## k = 6 : 1 = 11051 / 11052
## k = 7 : 1 = 11052 / 11052
## k = 8 : 1 = 11052 / 11052
```



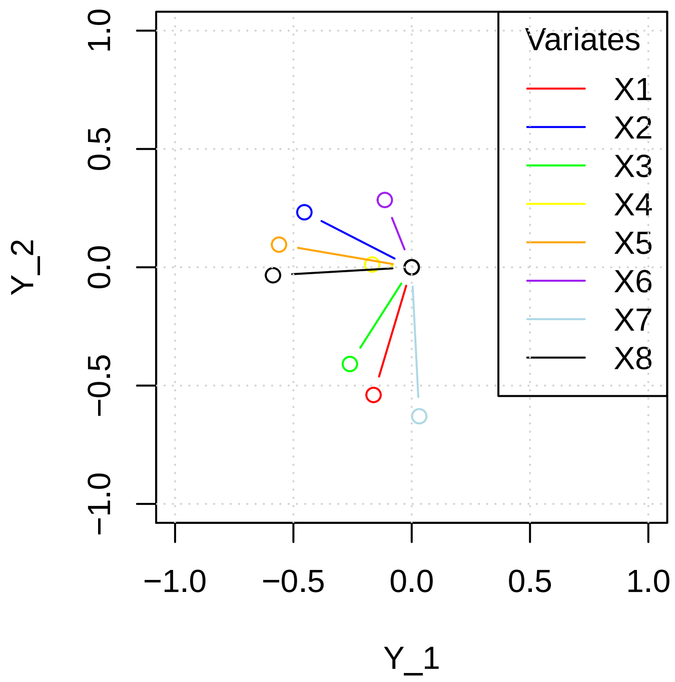
(1.e) Repeat with the data standarized. Aka, use R instead of S for the analysis.

```
plotProportionsTogether<-function(eigen_values, eigen_values_z) {
  ks <- c(0)
  props <- c(0)
  props_z <- c(0)
  RR <- sum(eigen_values)
  RR_z <- sum(eigen_values_z)

  for (k in 1:(r+1)) {
    ks <- c(ks, k)
    LL_r <- sum(eigen_values[1:k])
    props <- c(props, LL_r/RR)
    LL_r_z <- sum(eigen_values_z[1:k])
    props_z <- c(props_z, LL_r_z/RR_z)
  }

  plot(ks, props, col="red", type="b", xlab = 'Number of PCs used', ylab = 'Proportion of variance expl
  grid()
  points(ks, props_z, type="b", col="blue")
  legend('bottomright', title='Data', legend=c('Non-standarized', 'Standarized'),
       col=c('red','blue'), lty=1)
}

# Com
R <- cor(X)

# eigens
ev_z <- eigen(R)
eigen_values_z <- ev_z$values
V_z <- ev_z$vectors

#plotProportions(eigen_values_z, print=TRUE)
plotProportionsTogether(eigen_values, eigen_values_z)
```



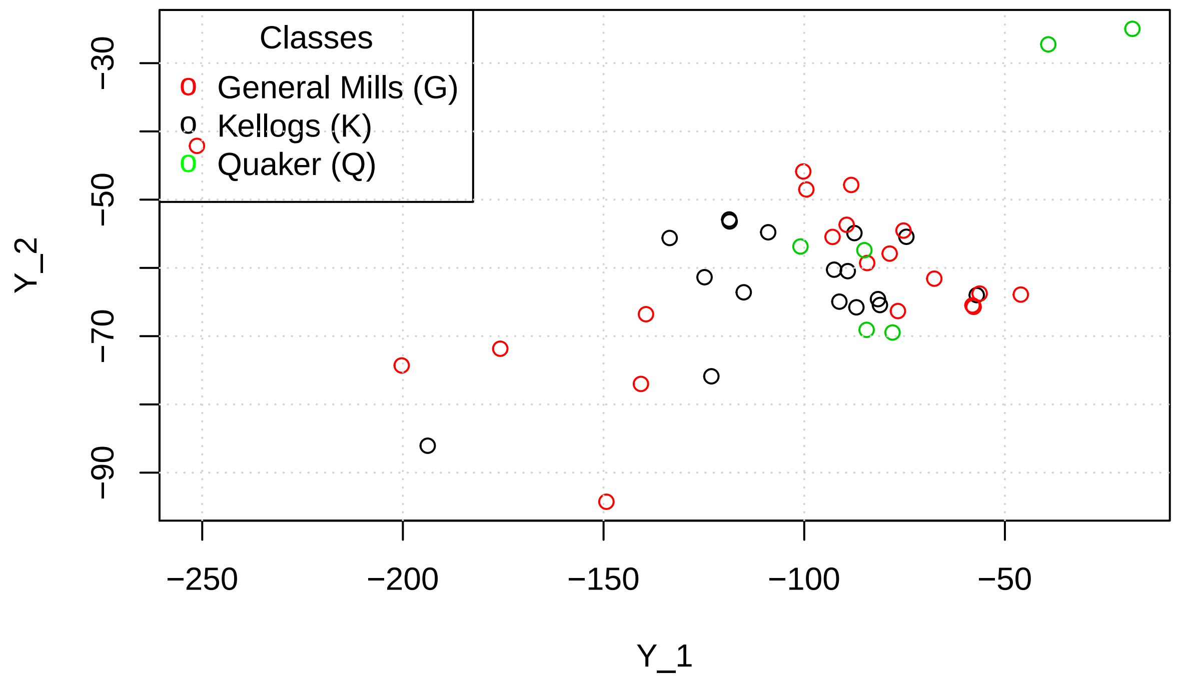
Plot PCs contributions

```
color <- c('red', 'blue', 'green', 'yellow', 'orange', 'purple', 'light blue', 'black')
par(pty="s")

for (i in 1:r) {
  vector_X <- c(0, V_z[i,1])
  vector_Y <- c(0, V_z[i,2])
  if (i==1)
    plot(vector_X, vector_Y, type='b', xlab='Y_1', ylab='Y_2', xlim=c(-1,1), ylim=c(-1,1), col=color[i]
  else
    lines(vector_X, vector_Y, type='b', col=color[i])
}
points(c(0), c(0))
legend("topright", legend=c('X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8'),
       col=color, lty=1, title="Variates")

grid()
```



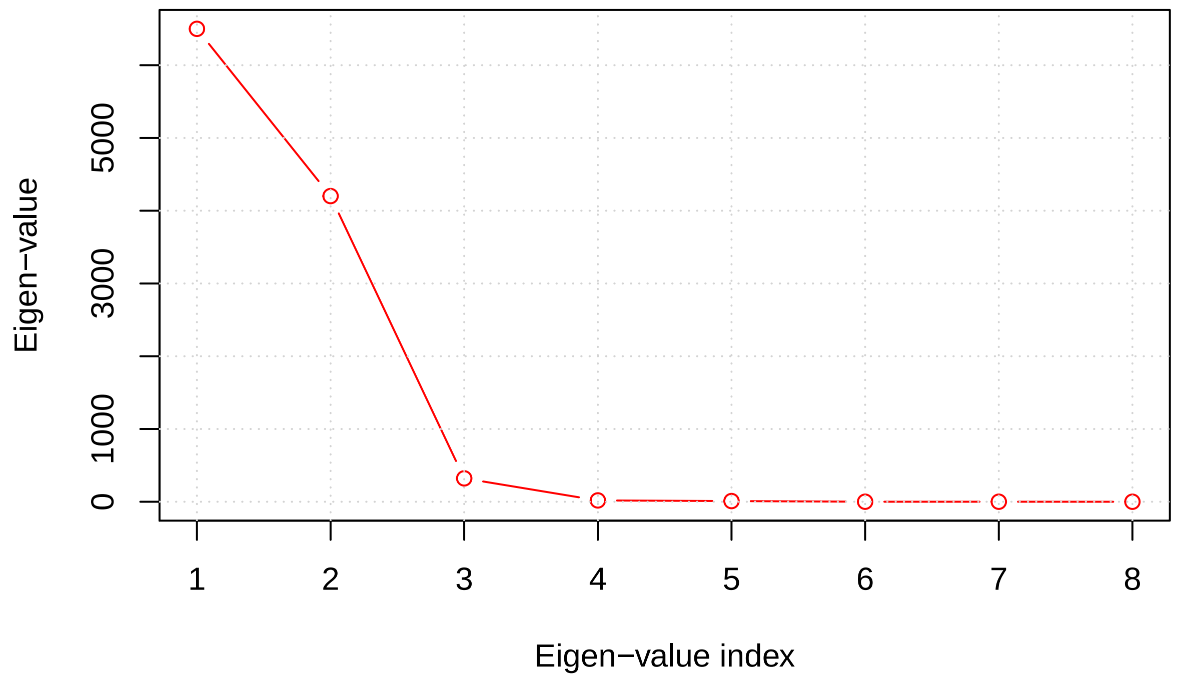
Plot PC scores

```
Y_scores <- X %*% V_z[,1:2]

#replace(classes, rep(length(classes)), c(rep('red',17), rep('blue',20), rep('green', 6)))
plot(Y_scores[,1], Y_scores[,2], xlab='Y_1', ylab='Y_2', col=classes)
legend("topleft", legend=c('General Mills (G)', 'Kellogs (K)', 'Quaker (Q)'),
       col=c('red', 'black', 'green'), pch='o', lty=0, title="Classes")
grid()
```

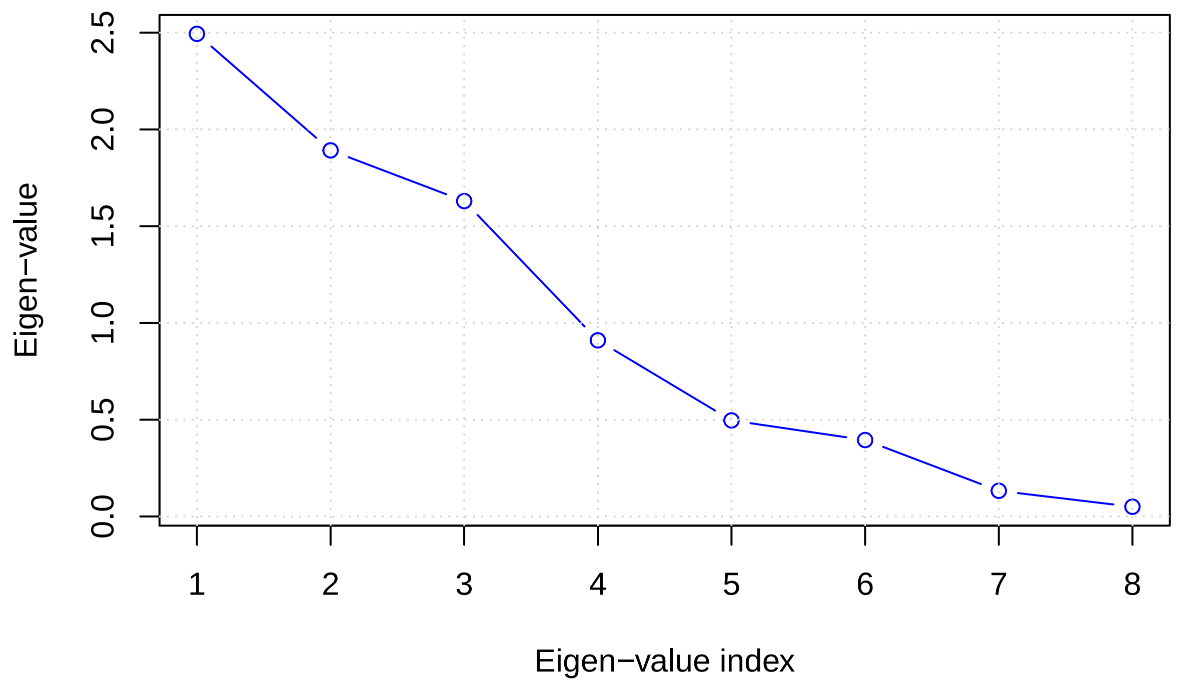


Plot eigen values

```
plot(rep(1:r), eigen_values, type='b', col='red', xlab='Eigen-value index', ylab='Eigen-value')
grid()
```



```
plot(rep(1:r), eigen_values_z, type='b', col='blue', xlab='Eigen-value index', ylab='Eigen-value')
grid()
```



Eigen−value
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7
8
Eigen−value index

# Final project: LDA

Jose Antonio Alvarez Ocete

For fractions:

```r
library(MASS)
options(digits=4)
```

```r
#import data
data <- read.table("data/anaconda.dat")

n1 <- 28
n2 <- 28

#set up X
X_1 <- as.matrix(data[1:n1,1:2])
X_2 <- as.matrix(data[(n1+1):(n1+n2),1:2])

names <- c('Snout Vent Length', 'Weight')
colnames(X_1)<- names
colnames(X_2) <- names

r <- length(names)

sampleMean<-function(X, n) {
  Ones <- rep(1,n)
  return (1/n * t(X)%*%Ones)
}

sampleCovariance<-function(X, n, sample_mean) {
  Ones <- rep(1,n)
  return (1/(n-1) * t(X - Ones%*%t(sample_mean))%*%(X - Ones%*%t(sample_mean)))
}
```

Conduct a two sample test to make sure the populations are different form each other.

```r
# Returns true if we reject the Ho: the populations are the same. Returns false if we fail to reject.
hotelling_test<-function(X_1, X_2, alpha=0.05, print=FALSE) {
  n1 <- length(X_1[,1])
  n2 <- length(X_2[,1])
  p <- length(X_1[1,])

  # Compute sample mean and S
  x1_sample_mean <- sampleMean(X_1, n1)
  x2_sample_mean <- sampleMean(X_2, n2)
  S1 <- sampleCovariance(X_1, n1, x1_sample_mean)
  S2 <- sampleCovariance(X_2, n2, x2_sample_mean)
  Spooled <- (n1-1)/(n1+n2-2) * S1 + (n2-1)/(n1+n2-2) * S2
  # Compute Hotelling's T^2 statistic
  diff <- x1_sample_mean - x2_sample_mean
  T_2 <- t(diff)%*%solve((1/n1 + 1/n2)*Spooled)%*%diff
  F <- (n1+n2-2)*p / (n1+n2-1-p) * qf(1-alpha, df1=p, df2=n1+n2-1-p)

  # Print the results if specified
  if (print) {
    cat('Reject if T_2 =', T_2, ' > ', F, '= F', fill=TRUE)
  }

  return (T_2 > F)
}

reject <- hotelling_test(X_1, X_2, alpha=0.05, print=TRUE)
```

```
## Reject if T_2 = 76.92  >  6.463 = F
```

```
cat('H_0: u_1 = u_2. Hypothesis rejected? --->', reject)
```

```
## H_0: u_1 = u_2. Hypothesis rejected? ---> TRUE
```

Classifier function

```
# If x_0 is NULL, the classifier is printed and nothing else is done
fisher<-function(X_1, X_2, x_0=NULL) {
  n1 <- length(X_1[,1])
  n2 <- length(X_2[,1])

  # Compute sample mean and S
  x1_sample_mean <- sampleMean(X_1, n1)
  x2_sample_mean <- sampleMean(X_2, n2)
  S1 <- sampleCovariance(X_1, n1, x1_sample_mean)
  S2 <- sampleCovariance(X_2, n2, x2_sample_mean)
  Spooled <- (n1-1)/(n1+n2-2) * S1 + (n2-1)/(n1+n2-2) * S2

  # Compute the classification
  w <- t(x1_sample_mean - x2_sample_mean) %*% solve(Spooled)
  mid <- 1/2*(x1_sample_mean + x2_sample_mean)

  # Print our classifier or classify the value
  if (is.null(x_0))
    cat(w, '*(x_0 -', mid, ') >= 0')
  else {
    result <- w%*%(x_0 - mid)
    if (result >= 0) {
      return (1)
    }
    return (2)
  }
}
```

Print the classiffier obtained

```
# Use the classifier with our whole population, without predicting anything
fisher(X_1, X_2)
```

```
## 0.05095 -0.01986 *(x_0 - 288.5 22.28 ) >= 0
```

Compute the apparent error rate (AER):

```
# Returns 0 if then prediction is correct, 1 otherwise.
classify<-function(X_1, X_2, x_0, expeted_class) {
  if ( fisher(X_1, X_2, x_0) != expeted_class ) {
    return (1)
  }
  return (0)
}

# Use the classifier with our whole population, without predicting anything
errors <- 0
for (i in 1:n1) {
  errors <- errors + classify(X_1, X_2, X_1[i,], 1)
}
for (i in 1:n2) {
  errors <- errors + classify(X_1, X_2, X_2[i,], 2)
}
AER <- errors / (n1+n2)
cat( fractions(AER), '=', errors, '/', (n1+n2))
```

```
## 0.08929 = 5 / 56
```

Compute the expected actual error rate (EAER):

```
# Use the classifier with our whole population, without predicting anything
errors <- 0
for (i in 1:n1) {
  errors <- errors + classify(X_1[-i,], X_2, X_1[i,], 1)
}
for (i in 1:n2) {
  errors <- errors + classify(X_1, X_2[-i,], X_2[i,], 2)
}
EAER <- errors / (n1+n2)
cat( fractions(EAER), '=', errors, '/', (n1+n2))
```

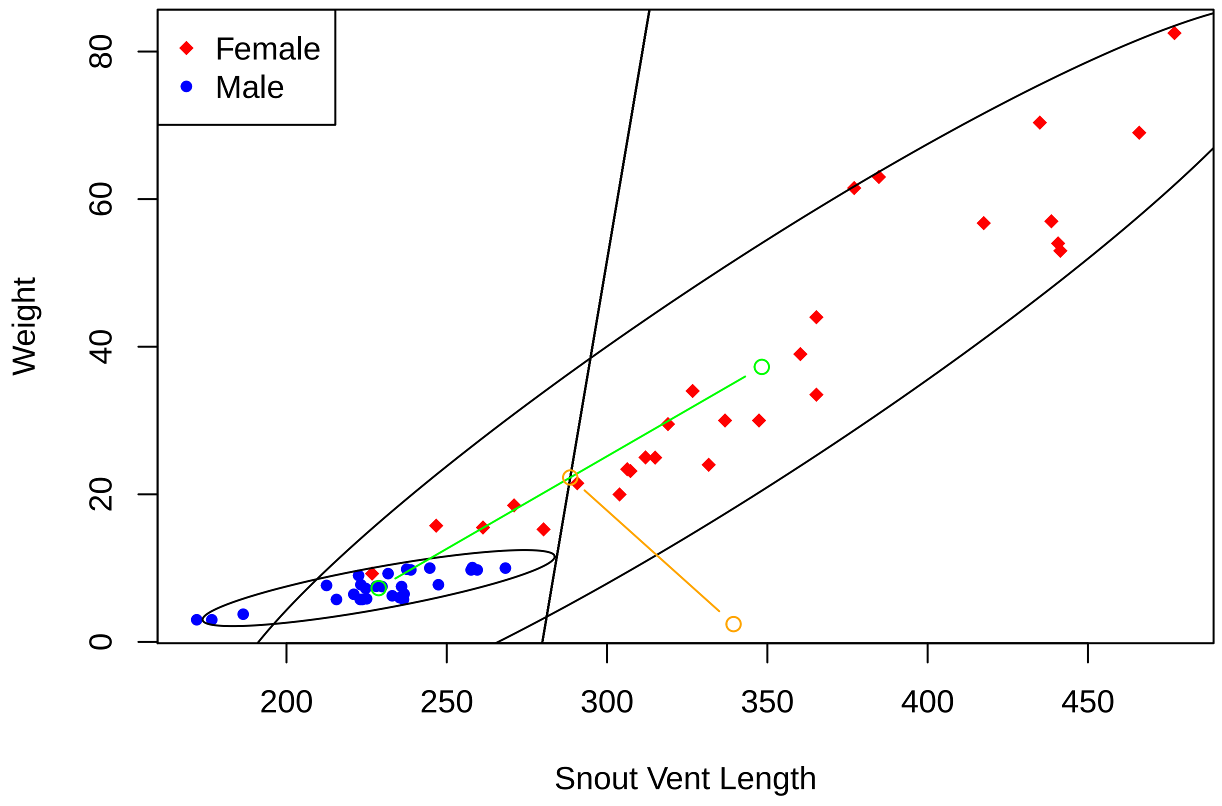
```
## 0.08929 = 5 / 56
```

```
#library(rrcov)
library(mixtools)
```

```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

```
par(mfrow=c(1,1), mar=c(4,4,2,1))
plot(data$V1,data$V2, xlab=names[1] ,ylab=names[2],
     pch=rep(c(18,20),each=28), col=rep(c(2,4), each=28), main="")
legend("topleft", legend=c("Female","Male"), pch=c(18,20), col=c(2,4), cex=1)

# Method 1
x1 <- X_1
x2 <- X_2

# compute sample mean vectors:
x1.mean <- colMeans(x1)
x2.mean <- colMeans(x2)
```

```
# compute pooled estimate for the covariance matrix:
S.u <- (n1-1)/(n1+n2-2) * var(x1) + (n2-1)/(n1+n2-2) * var(x2)
w <- solve(S.u)%*%(x1.mean-x2.mean)
w0 <- -(x1.mean+x2.mean)%*%w/2

lines(data[,1],-(w[1]*data[,1]+c(w0))/w[2])
lines(cbind(x1.mean, x2.mean))

alpha <- .05
ellipse(x1.mean, var(x1), alpha=alpha, npoints=250, newplot=FALSE)
ellipse(x2.mean, var(x2), alpha=alpha, npoints=250, newplot=FALSE)

# Compute line between mean points and the line w
points(c(x2.mean[1],x1.mean[1]), c(x2.mean[2],x1.mean[2]), type='b', col='green')
mid <- (x1.mean + x2.mean)/2

new_w <- 1000*w + mid

points(c(new_w[1], mid[1]), c(new_w[2], mid[2]), type='b', col='orange')
```


Female
Male
Weight
Snout Vent Length