Project 2

Blake Pappas

December 19, 2023

Problem 1

In this problem, we are going to use ERA-Interim data product to conduct a PCA analysis. The ERA-Interim is a global atmospheric reanalysis dataset. Reanalysis is an approach to produce spatially and temporally gridded datasets via data assimilation for climate monitoring and analysis. In the R data file `NA_MaxT ERA.RData`, you will find daily values of maximum temperature from 1979 to 2017 with the spatial resolution around 80km. The original global extent has been subsetted to a region covering contiguous United States and extending into Canada and Mexico.

1. First, use the following R script to load the R data object:

```
load("NA_MaxT_ERA.RData")
NA_MaxT <- NA_MaxT - 273.15 # Converts from Kelvin to Celsius
nlon <- length(NA_lon) # Number of longitude grid points
nlat <- length(NA_lat) # Number of latitude grid points
nmon <- 12 # Number of months
nyr <- 39 # Number of years
```

2. Second, aggregate the daily data to monthly average:

```
avg_by_mon <- array(dim = c(nlon, nlat, nmon, nyr))
for (i in 1:nlon) {
  for (j in 1:nlat) {
dat <- cbind(NA_MaxT[i, j, ], as.factor(mon), as.factor(yr))
avg_by_mon[i, j, , ] <- tapply(dat[, 1], list(dat[, 2], dat[, 3]), mean)
  }
}
```

3. Third, compute the "anomalies" by subtracting the 39-year monthly average at each location:

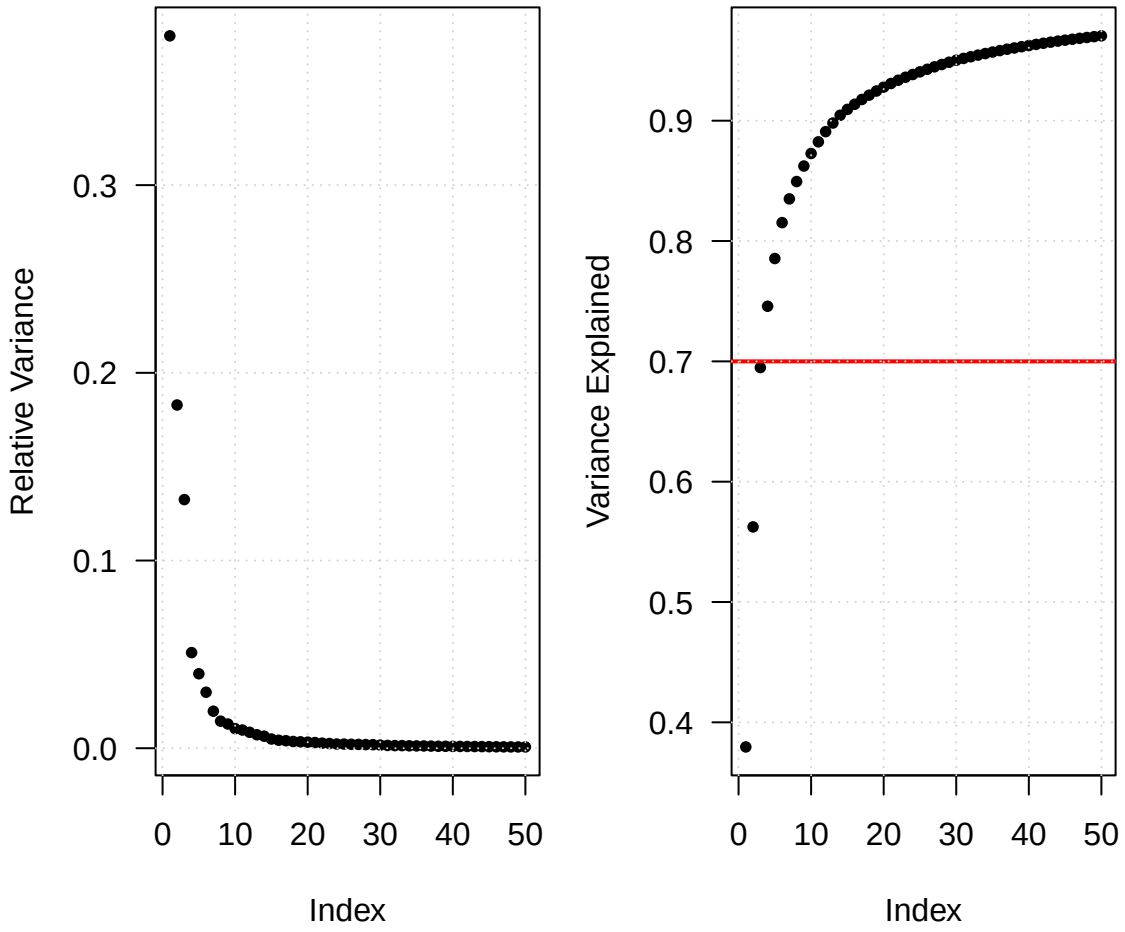
```
maxT_temp <- apply(avg_by_mon, 1:3, function(x) x - mean(x, na.rm = T))

# Change the Data Into Longitude-Latitude-Month Format
maxT_anomalies <- array(dim = c(nlon, nlat, nmon * nyr))
for (i in 1:nlon) {
  for (j in 1:nlat) {
    maxT_anomalies[i, j, ] <- c(t(maxT_temp[, i, j, ]))
  }
}
```

4. Fourth, apply singular value decomposition to perform principal component analysis:

```
temp <- array(maxT_anomalies, c(nlon * nlat, nmon * nyr))

# Convert the Data Into a Row/Column Format
temp2 <- svd(temp)
```

(a) How many principal components are needed in order to explain more than 70% of the variation in anomalies?

```
# Screen Plots for EOFs

# Relative Variance Plot
par(mar = c(4, 4, 1, 1), mfrow = c(1, 2), las = 1)
dt <- ((temp2$d^2) / sum(temp2$d^2))
plot(1:50, dt[1:50], xlab = "Index", ylab = "Relative Variance",
     pch = 16, cex = 0.8)
grid()

# Variance Explained Plot
dt <- (cumsum(temp2$d^2) / sum(temp2$d^2))
plot(1:50, dt[1:50], xlab = "Index", ylab = "Variance Explained", pch = 16, cex = 0.8)
abline(h = 0.7, col = "red", lwd = 2)
grid()
```



Answer: 4 principal components are required in order to explain more than 70% of the variation in anomalies. Looking at the screen plots, the fourth principal component puts the variation explained at approximately 74.58%.

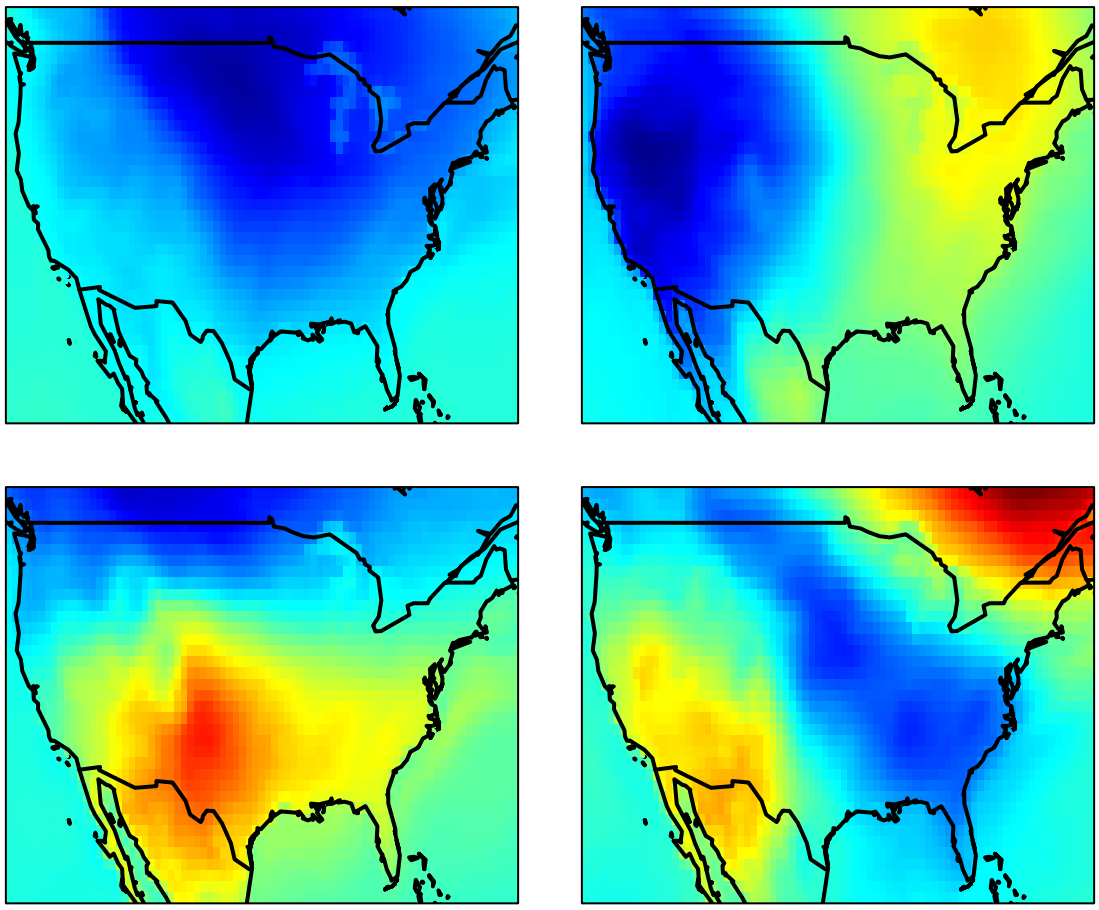
(b) Based on the answer in (a), plot these principal components in terms of spatial fields and try to interpret these principal components.

```
library(fields)
library(maps)

temp <- array(maxT_anomalies, c(nlon * nlat, nmon * nyr))
ind <- is.na(temp[, 1])
temp <- temp[!ind, ]
temp2 <- svd(temp)
U1 <- matrix(NA, nlon * nlat)
U1[!ind] <- temp2$u[, 1]; U1 <- matrix(U1, nlon, nlat)
U2 <- matrix(NA, nlon * nlat)
U2[!ind] <- temp2$u[, 2]; U2 <- matrix(U2, nlon, nlat)
U3 <- matrix(NA, nlon * nlat)
U3[!ind] <- temp2$u[, 3]; U3 <- matrix(U3, nlon, nlat)
U4 <- matrix(NA, nlon * nlat)
U4[!ind] <- temp2$u[, 4]; U4 <- matrix(U4, nlon, nlat)
zr <- range(c(U1, U2, U3, U4), na.rm = TRUE)
```

```r
set.panel(2, 2)
```

```
## plot window will lay out plots in a 2 by 2 matrix
```

```r
par(oma = c(0, 0, 0, 0))
ct <- tim.colors(256)
par(mar = c(1, 1, 1, 1))
image(NA_lon, NA_lat, U1, axes = FALSE, xlab = "", ylab = "", zlim = zr, col = ct)
map("world", add = TRUE, lwd = 2)
box()
image(NA_lon, NA_lat, U2, axes = FALSE, xlab = "", ylab = "", zlim = zr, col = ct)
map("world", add = TRUE, lwd = 2)
box()
image(NA_lon, NA_lat, U3, axes = FALSE, xlab = "", ylab = "", zlim = zr, col = ct)
map("world", add = TRUE, lwd = 2)
box()
image(NA_lon, NA_lat, U4, axes = FALSE, xlab = "", ylab = "", zlim = zr, col = ct)
map("world", add = TRUE, lwd = 2)
box()
```



Principal component #1 seems to exhibit a land-ocean contrast, as spatial regions closer to the ocean tend to have higher temperatures than spatial regions closer to the interior of the contiguous United States, Canada,

and Mexico. This finding makes sense, as water is less resistant to changes in temperature than air. In general, the closer that a spatial region is to a large mass of water, the less responsive that region is to changes in temperature, in comparison to spatial regions that are geographically distant from large masses of water. Principal component #2 appears to exhibit an east-west contrast, as the monthly temperatures in the east are higher than the monthly temperatures in the west. Principal component #3 appears to exhibit a north-south contrast, as the monthly temperatures in the north are lower than the monthly temperatures in the south. Principal component #4 appears to exhibit a southwest-interior-northeast contrast, as the monthly temperatures in the southwest and northeast are higher than the monthly temperatures in the interior.

Problem 2

Split the decathlon data into two different sets:

- X: Shot.put, Discus, Javeline, Pole.vault
- Y : 100m, 400m, 1500m, 110m.hurdle, Long.jump, High.jump

and perform a canonical correlation analysis.

The data comes from two sporting events in 2004: The Olympics and Decastar. This sample contains the results of 41 decathletes in 10 track and fields events in an attempt to determine which factors influence performance. In the analysis, two collections of variables were measured:

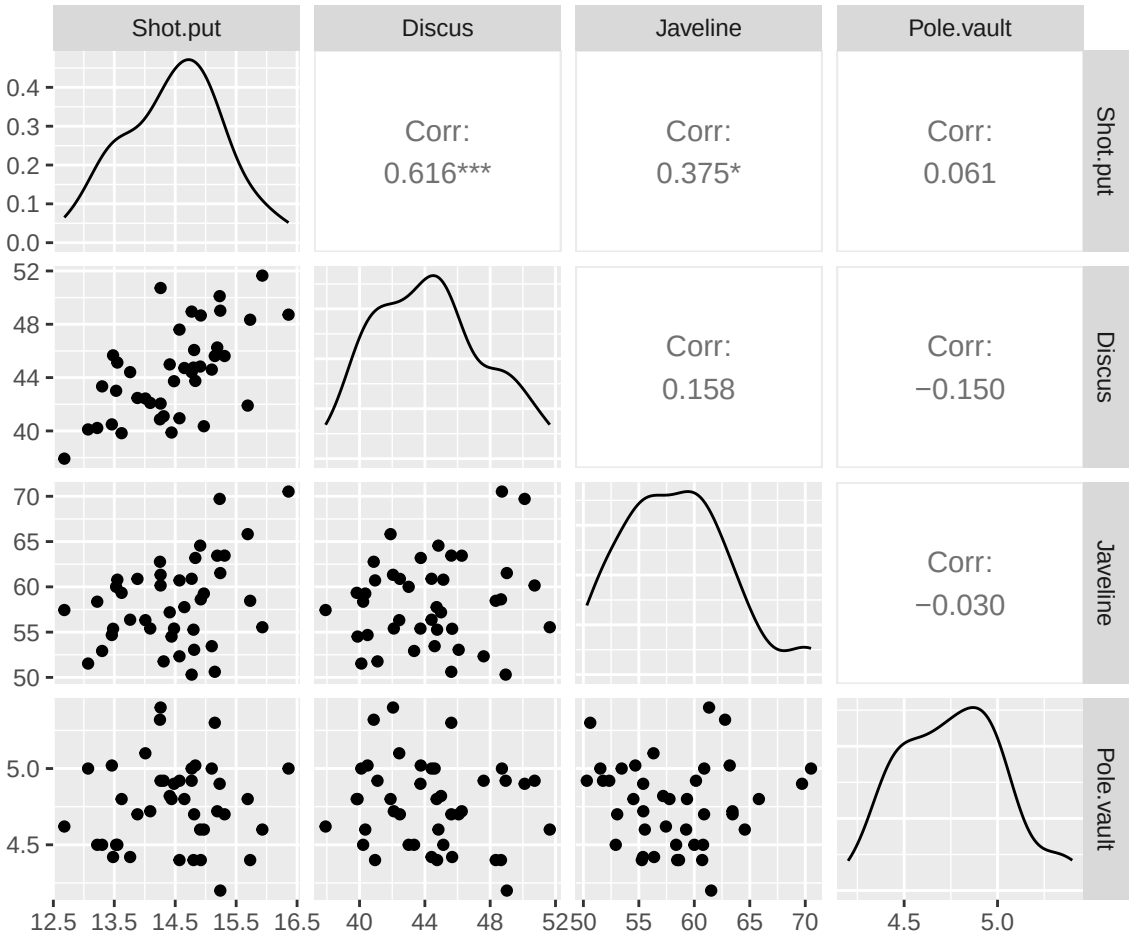
- Field Events: Shot Put, Discus, Javelin, Pole Vault
- Track Events: 100-Meter Dash, 400-Meter Dash, 1500-Meter Run, 110-Meter Hurdles, Long Jump, High Jump

Load the Data and Import All Appropriate Packages

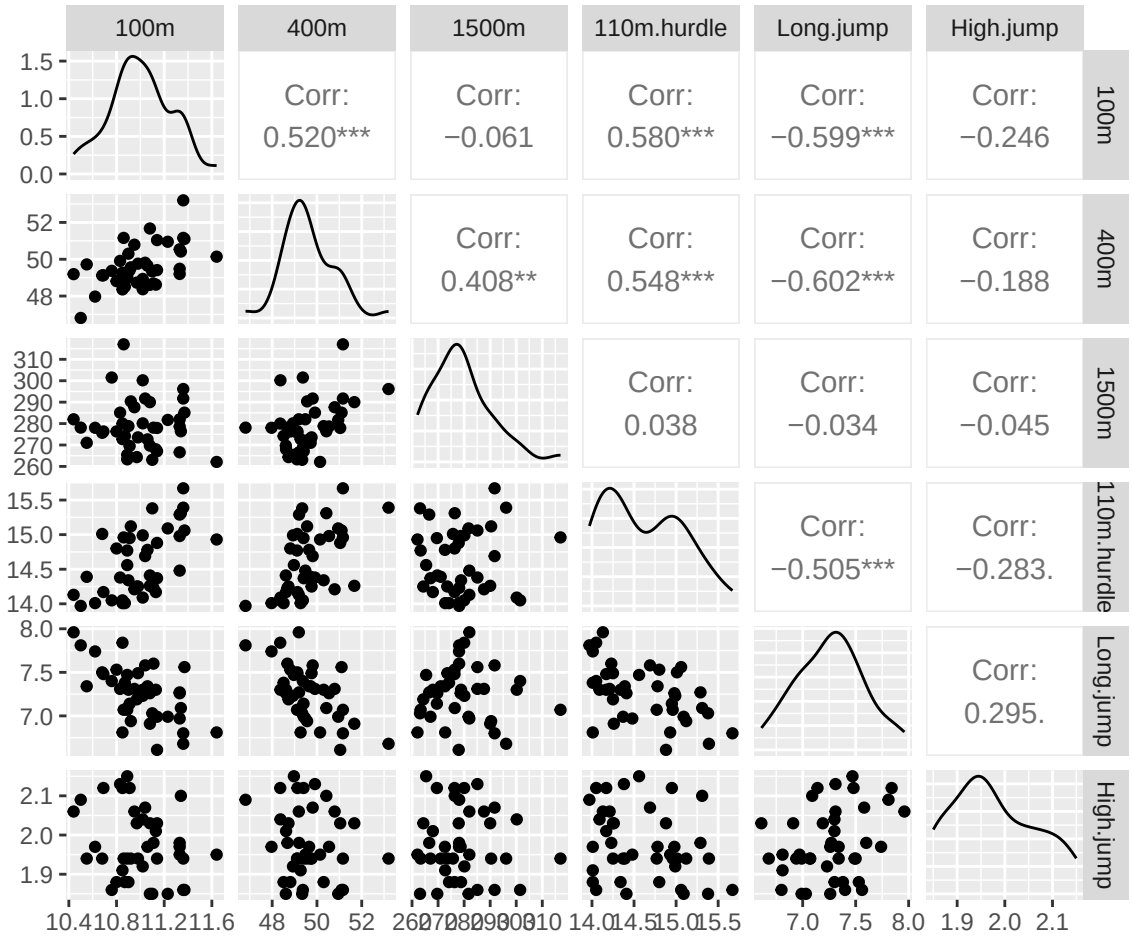
```
Packages <- c("ggplot2", "GGally", "ellipse", "RColorBrewer",
              "CCA", "CCP")
lapply(Packages, library, character.only = TRUE)
library(tidyverse)
library(FactoMineR)
data(decathlon)
```

Summarizing *X (Field Events)* and *Y (Track Events)*

```
# Subset the Data Into Two Sets: X (Field Events) and Y (Track Events)
X <- decathlon[, c(3, 7, 9, 8)]
Y <- decathlon[, c(1, 5, 10, 6, 2, 4)]

# Shows the Scatterplot Matrix of the X (Field Events) Subset
ggpairs(X)
```




```
# Shows the Scatterplot Matrix of the Y (Track Events) Subset
ggpairs(Y)
```




```
# Calculate the Cross-Covariance Structure
matcor(X, Y)
```

```
## $Xcor
##               Shot.put     Discus   Javeline  Pole.vault
## Shot.put   1.00000000  0.6157681  0.3749555  0.06118185
## Discus     0.61576810  1.0000000  0.1578898 -0.15007240
## Javeline   0.37495551  0.1578898  1.0000000 -0.03000060
## Pole.vault 0.06118185 -0.1500724 -0.0300006  1.00000000
##
## $Ycor
##                     100m       400m       1500m 110m.hurdle   Long.jump
## 100m         1.00000000  0.5202982 -0.06054645  0.57988893 -0.59867767
## 400m         0.52029815  1.0000000  0.40810643  0.54798776 -0.60206262
## 1500m       -0.06054645  0.4081064  1.00000000  0.03754024 -0.03368613
## 110m.hurdle  0.57988893  0.5479878  0.03754024  1.00000000 -0.50541009
## Long.jump   -0.59867767 -0.6020626 -0.03368613 -0.50541009  1.00000000
## High.jump   -0.24625292 -0.1879569 -0.04490252 -0.28328909  0.29464444
##                High.jump
## 100m        -0.24625292
## 400m        -0.18795693
## 1500m       -0.04490252
## 110m.hurdle -0.28328909
```

```
## Long.jump    0.29464444
## High.jump    1.00000000
##
## $XYcor
##                 Shot.put     Discus     Javeline   Pole.vault        100m
## Shot.put     1.00000000  0.6157681  0.374955509  0.061181853 -0.35648227
## Discus       0.61576810  1.0000000  0.157889799 -0.150072400 -0.22170757
## Javeline     0.37495551  0.1578898  1.000000000 -0.030000603 -0.15774645
## Pole.vault   0.06118185 -0.1500724 -0.030000603  1.000000000 -0.08253683
## 100m        -0.35648227 -0.2217076 -0.157746452 -0.082536834  1.00000000
## 400m        -0.13843292 -0.1178794  0.004232096 -0.079292469  0.52029815
## 1500m        0.11580306  0.2581751 -0.180393128  0.247447780 -0.06054645
## 110m.hurdle -0.25161571 -0.3262010  0.008743251 -0.002703885  0.57988893
## Long.jump    0.18330436  0.1943101  0.119758933  0.204014112 -0.59867767
## High.jump    0.48921153  0.3692183  0.171880092 -0.156180742 -0.24625292
##                     400m       1500m  110m.hurdle   Long.jump   High.jump
## Shot.put    -0.138432919  0.11580306 -0.251615714  0.18330436  0.48921153
## Discus      -0.117879365  0.25817510 -0.326200961  0.19431009  0.36921834
## Javeline     0.004232096 -0.18039313  0.008743251  0.11975893  0.17188009
## Pole.vault  -0.079292469  0.24744778 -0.002703885  0.20401411 -0.15618074
## 100m         0.520298155 -0.06054645  0.579888931 -0.59867767 -0.24625292
## 400m         1.000000000  0.40810643  0.547987756 -0.60206262 -0.18795693
## 1500m        0.408106432  1.00000000  0.037540240 -0.03368613 -0.04490252
## 110m.hurdle  0.547987756  0.03754024  1.000000000 -0.50541009 -0.28328909
## Long.jump   -0.602062618 -0.03368613 -0.505410086  1.00000000  0.29464444
## High.jump   -0.187956928 -0.04490252 -0.283289090  0.29464444  1.00000000
```

Likelihood Ratio Test

Next, determine if there is any relationship between the two sets of variables.

```
# Tests of Canonical Dimensions
rho <- cc(X, Y)$cor

# Define the Number of Observations
n <- dim(X)[1]

# Define the Number of Variables in First Set
p <- length(X)

# Define the Number of Variables in the Second Set
q <- length(Y)

# Calculate the p-Values Using the F-Approximations of Different Test Statistics
CCA_Test <- p.asym(rho, n, p, q, tstat = "Wilks")
```

```
## Wilks' Lambda, using F-approximation (Rao's F):
##               stat    approx df1       df2   p.value
## 1 to 4:  0.3616269 1.5424598  24 109.35612 0.0690605
## 2 to 4:  0.5740891 1.3172953  15  88.73931 0.2090087
## 3 to 4:  0.8772043 0.5585326   8  66.00000 0.8077719
## 4 to 4:  0.9668859 0.3881464   3  34.00000 0.7622450
```

Hypothesis Test:

H_0: $\rho_1 = \rho_2 = \rho_3 = \rho_4 = 0$
H_A: $\rho_1 \neq \rho_2 \neq \rho_3 \neq \rho_4 \neq 0$
Confidence Level: $\alpha = 0.05$
Test Statistic : F = 1.5424598
p-value: 0.0690605
Conclusion: Fail to Reject H_0

H_0: $\rho_2 = \rho_3 = \rho_4 = 0$
H_A: $\rho_2 \neq \rho_3 \neq \rho_4 \neq 0$
Confidence Level: $\alpha = 0.05$
Test Statistic : F = 1.3172953
p-value: 0.2090087
Conclusion: Fail to Reject H_0

H_0: $\rho_3 = \rho_4 = 0$
H_A: $\rho_3 \neq \rho_4 \neq 0$
Confidence Level: $\alpha = 0.05$
Test Statistic : F = 0.5585326
p-value: 0.8077719
Conclusion: Fail to Reject H_0

H_0: $\rho_4 = 0$
H_A: $\rho_4 \neq 0$
Confidence Level: $\alpha = 0.05$
Test Statistic : F = 0.3881464
p-value: 0.762245
Conclusion: Fail to Reject H_0

Based on the hypothesis testing, the cross covariance between variables is equal to zero, which means that all four canonical variate pairs are not significantly correlated and dependent on one another. This suggests that we may not summarize all four pairs. Therefore, there is no need to pursue canonical correlation analysis.

However, for the fun of it, let's assume they are significantly correlated and interdependent and conduct the canonical correlation analysis.

Estimates of Canonical Correlation

```
# Estimates of Canonical Correlation
cc1 <- cc(X, Y)

# Canonical Correlation
cc1$cor
```

```
## [1] 0.6083468 0.5878324 0.3045538 0.1819728
```

```
# Coefficient of Determination (R-Squared)
cc1$cor^2
```

```
## [1] 0.37008581 0.34554692 0.09275300 0.03311412
```

37.01% of the variation in U_1 is explained by the variation in V_1, 34.55% of the variation in U_2 is explained by V_2, 9.28% of the variation in U_3 is explained by V_3, and only 3.31% of the variation in U_4 is explained by V_4.

Obtain the Canonical Coefficients

```
colnames(cc1$xcoef) <- c("U1", "U2", "U3", "U4")
cc1$xcoef
```

```
##                    U1          U2          U3         U4
## Shot.put    0.37562679 -1.17416698 -0.07837495 -1.1488183
## Discus      0.22026110  0.13719533  0.02605206  0.2894322
## Javeline   -0.09400791 -0.03665866 -0.14716835  0.1377135
## Pole.vault  1.29649330  2.35442844 -2.49620842 -0.6443607
```

The first canonical variable for X (Field Events) is
$U_1 = \mathbf{0.37562679X_{Shot.put} + 0.22026110X_{Discus} - 0.09400791X_{Javeline} + 1.29649330X_{Pole.vault}}$

```
colnames(cc1$ycoef) <- c("V1", "V2", "V3", "V4")
cc1$ycoef
```

```
##                    V1          V2          V3          V4
## 100m         0.6428069  2.72395703  2.321453493  2.44468195
## 400m        -0.4748815 -0.32424940 -0.340589632  0.89788924
## 1500m        0.0784419  0.04473333  0.008766086 -0.03748802
## 110m.hurdle -0.6540028 -0.13610668 -1.364777812 -1.87456121
## Long.jump   -0.3490155  1.44186376 -3.037473973  2.71466150
## High.jump    4.5505653 -8.58007729  1.570999493  0.54820741
```

The first canonical variable for Y (Track Events) is
$V_1 = \mathbf{0.6428069Y_{100m} - 0.4748815Y_{400m} + 0.0784419Y_{1500m} - 0.6540028Y_{110m.hurdle} - 0.3490155_{Long.jump} + 4.5505653Y_{High.jump}}$

Correlations Between Each Variable and the Corresponding Canonical Variate

Correlations Between X's and U's

```
colnames(cc1$scores$corr.X.xscores) <- c("U1", "U2", "U3", "U4")
cc1$scores$corr.X.xscores
```

```
##                   U1         U2         U3         U4
## Shot.put    0.6197254 -0.7089549 -0.3192355 -0.1068276
## Discus      0.8089633 -0.2588136  0.0401965  0.5262871
## Javeline   -0.2309859 -0.4863740 -0.6998697  0.4693275
## Pole.vault  0.2813295  0.5310669 -0.6897940 -0.4037398
```

Correlations Between Y's and V's

```
colnames(cc1$scores$corr.Y.xscores) <- c("V1", "V2", "V3", "V4")
cc1$scores$corr.Y.xscores
```

```
##                    V1          V2          V3          V4
## 100m        -0.2335160  0.21622530  0.17285577  0.03080478
```

```
## 400m        -0.1610720  0.02672888  0.05058977  0.03288368
## 1500m        0.3989873  0.20142726 -0.02833528 -0.02150894
## 110m.hurdle -0.3255576  0.08908164 -0.01678207 -0.07430677
## Long.jump    0.2205234  0.02494930 -0.22139092  0.05941827
## High.jump    0.2919154 -0.43509809 -0.01283400  0.03985680
```

Correlations Between Each Set of Variables and the Opposite Group of Canonical Variates

Correlations Between X's and V's

```
colnames(cc1$scores$corr.X.yscores) <- c("V1", "V2", "V3", "V4")
cc1$scores$corr.X.yscores
```

```
##                    V1         V2          V3          V4
## Shot.put    0.3770079 -0.4167467 -0.09722437 -0.01943973
## Discus      0.4921302 -0.1521390  0.01224200  0.09576996
## Javeline   -0.1405195 -0.2859064 -0.21314795  0.08540486
## Pole.vault  0.1711459  0.3121783 -0.21007936 -0.07346969
```

Correlations Between Y's and U's

```
colnames(cc1$scores$corr.Y.yscores) <- c("U1", "U2", "U3", "U4")
cc1$scores$corr.Y.yscores
```

```
##                     U1          U2          U3         U4
## 100m        -0.3838534  0.36783496  0.56757061  0.1692823
## 400m        -0.2647701  0.04547024  0.16611113  0.1807065
## 1500m        0.6558550  0.34266104 -0.09303868 -0.1181986
## 110m.hurdle -0.5351513  0.15154259 -0.05510378 -0.4083399
## Long.jump    0.3624963  0.04244288 -0.72693541  0.3265227
## High.jump    0.4798503 -0.74017372 -0.04214034  0.2190261
```

Problem 3

Perform a classification analysis using the Swiss bank notes data. Split the data into "training" and "testing" sets to evaluate the performance.

Load the Data

```
dat = read.csv("swiss3.csv", header = FALSE, skip = 1)
```

Create the Training and Testing Sets

```
sample_size <- floor(2 / 3 * nrow(dat))

set.seed(123)
training_indicator <- sample(seq_len(nrow(dat)), size = sample_size)
```

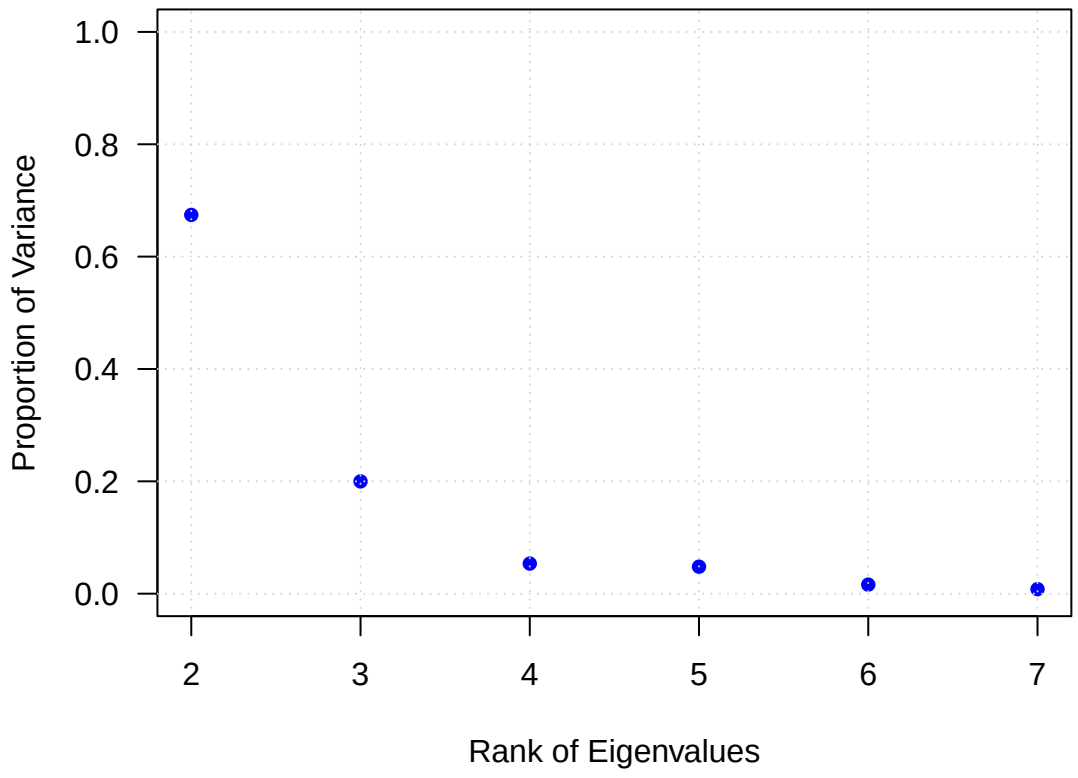
```r
training <- dat[training_indicator, ] # 137 observations
testing <- dat[-training_indicator, ] # 67 observations
```

I decided to create a training set that was 2/3's (137 observations) of the data and a testing set that was 1/3's of the data (67 observations).

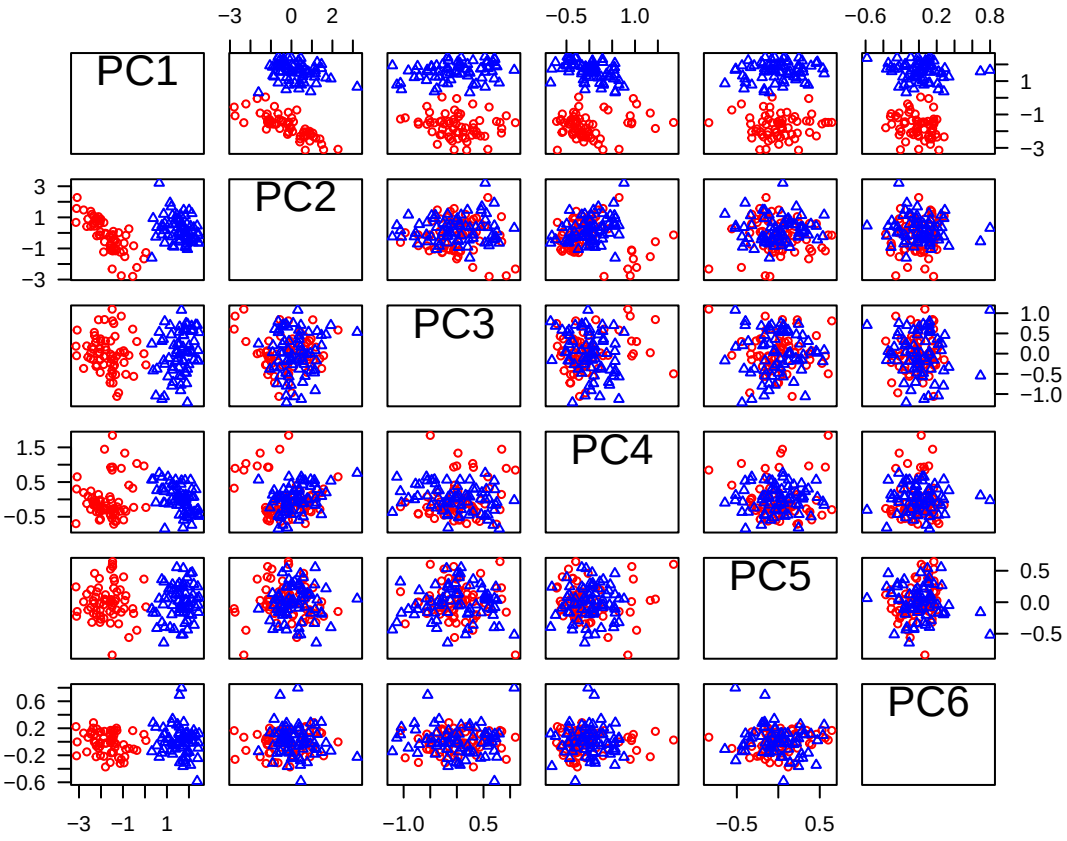
Training Set

Principal Component Analysis (PCA)

```r
# PCA
library(car)
training_type <- factor(training$V1)
pca <- prcomp(training[, 2:7])
Z <- pca$x
lambda <- pca$sdev^2 # Eigenvalues
par(las = 1)
plot(2:7, lambda / sum(lambda), xaxt = "n", las = 1, xlab = "Rank of Eigenvalues",
     ylab = "Proportion of Variance", pch = 16, col = "blue", cex = 1, ylim = c(0, 1))
grid()
axis(1, at = 2:7)
```



```r
# Scatterplot Matrix
scatterplotMatrix(~ Z | training_type, col = c("red", "blue"), diagonal = F, smooth = F,
                  regLine = F, legend = F, cex = 0.75)
```



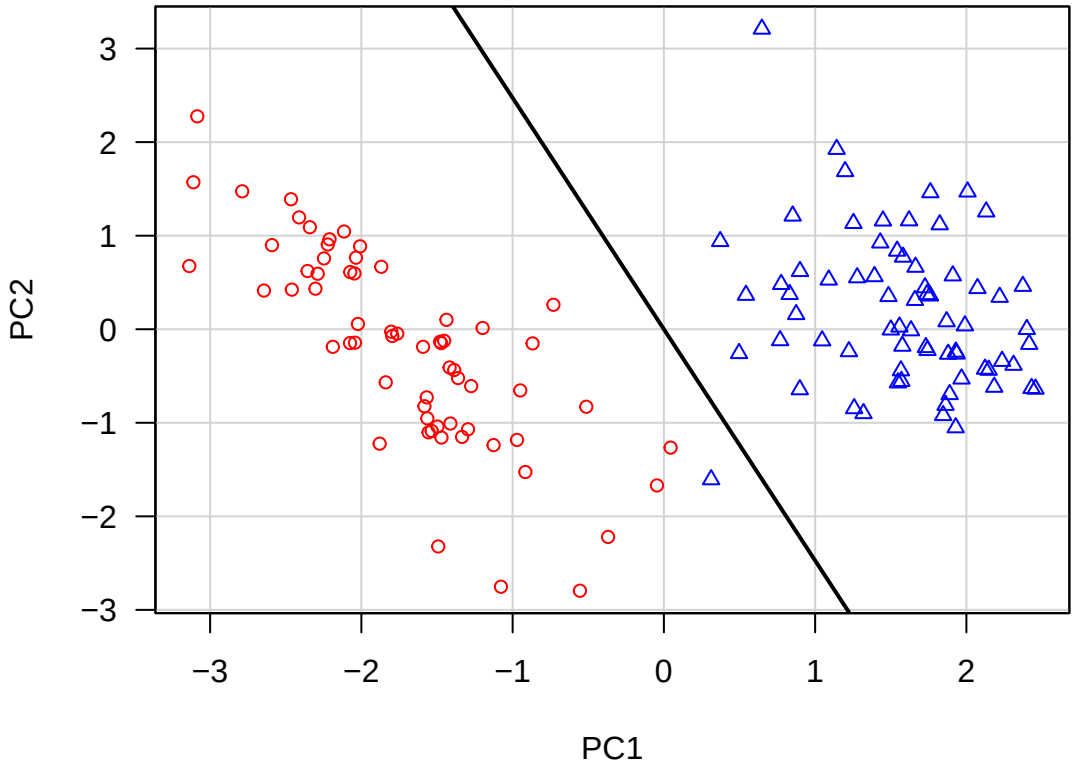
Linear Discriminant Analysis (LDA)

```r
library(MASS)

# LDA
fit <- lda(training_type ~ Z[, 1:2])
fit
```

```
## Call:
## lda(training_type ~ Z[, 1:2])
##
## Prior probabilities of groups:
##     fake     real
## 0.481203 0.518797
##
## Group means:
```

```
##      Z[, 1:2]PC1 Z[, 1:2]PC2
## fake   -1.697908  -0.2035927
## real    1.574871   0.1888396
##
## Coefficients of linear discriminants:
##                     LD1
## Z[, 1:2]PC1 1.9636746
## Z[, 1:2]PC2 0.7941343
```

```
par(las = 1)
scatterplot(PC2 ~ PC1 | training_type, Z, smooth = F, regLine = F, legend = F, cex = 0.85,
            col = c("red", "blue"))
abline(0, -fit$scaling[1] / fit$scaling[2], pch = 5, lwd = 2)
```

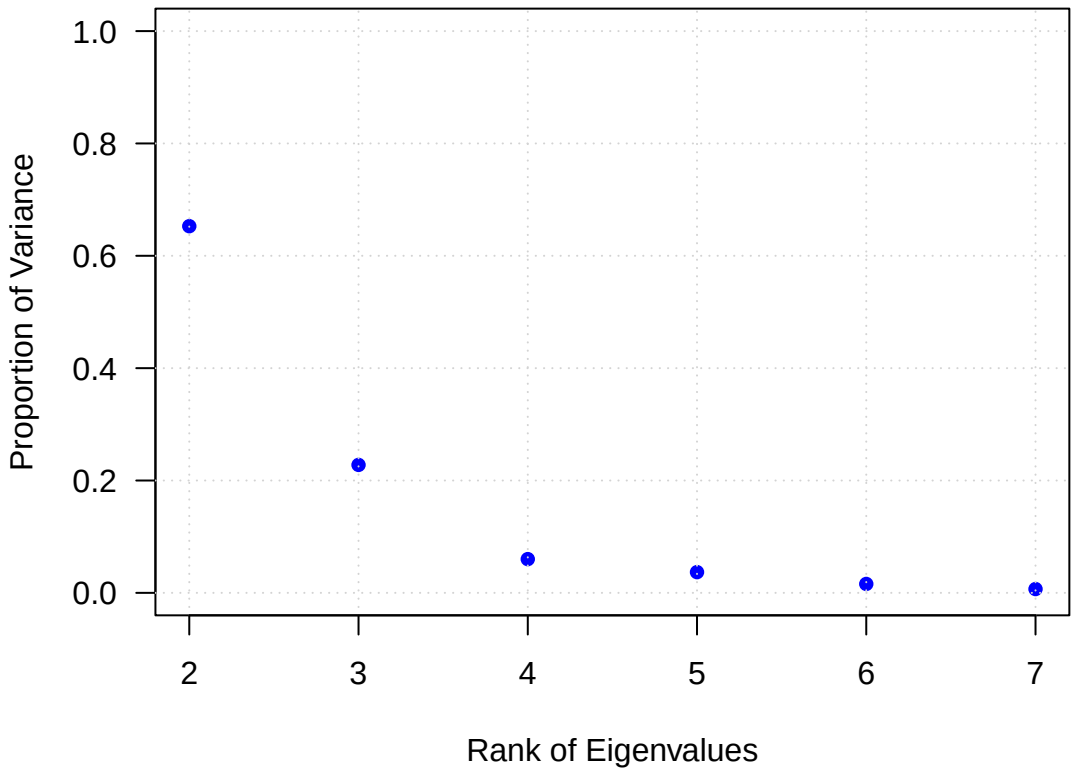


The Linear Discriminant Analysis training set shows that all the real bank notes (red circles) are correctly classified within the plot. However, the plotting of the fake bank notes (blue triangles) indicates that one was improperly classified. Despite the misclassification, the proper classification rate for the training set remains high and the chance of a type I error remains low. We can conclude that the training set model should be good to apply to the testing set.

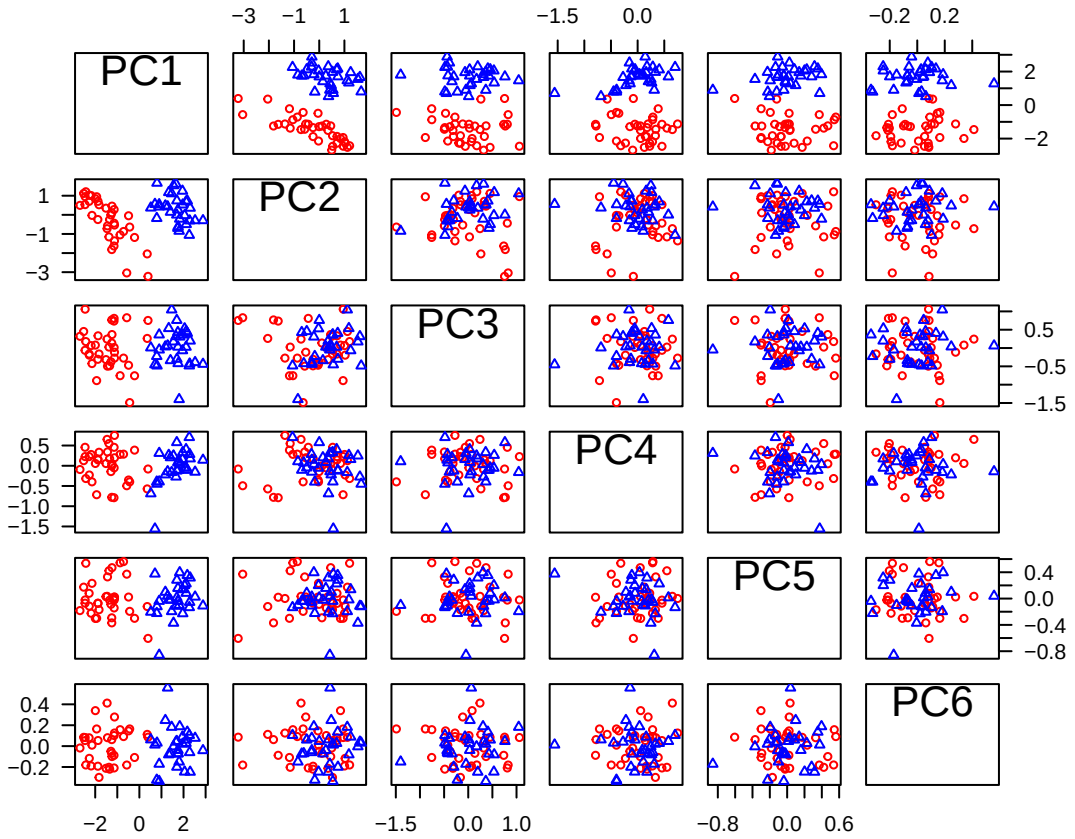
Testing Set

Principal Component Analysis (PCA)

```r
# PCA
testing_type <- factor(testing$V1)
pca <- prcomp(testing[, 2:7])
Z <- pca$x
lambda <- pca$sdev^2 # Eigenvalues
par(las = 1)
plot(2:7, lambda / sum(lambda), xaxt = "n", las = 1, xlab = "Rank of Eigenvalues",
     ylab = "Proportion of Variance", pch = 16, col = "blue", cex = 1, ylim = c(0, 1))
grid()
axis(1, at = 2:7)
```




```r
# Scatterplot Matrix
scatterplotMatrix(~ Z | testing_type, col = c("red", "blue"), diagonal = F, smooth = F,
                  regLine = F, legend = F, cex = 0.75)
```



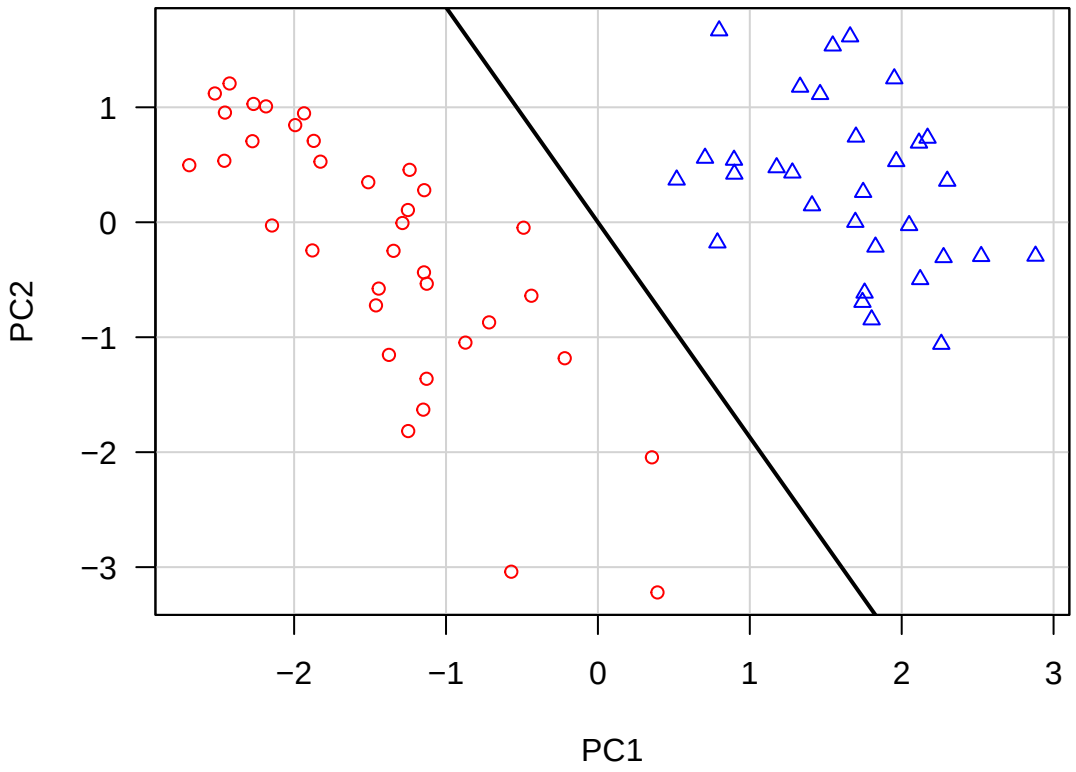
Linear Discriminant Analysis (LDA)

```
# LDA
fit <- lda(testing_type ~ Z[, 1:2])
fit
```

```
## Call:
## lda(testing_type ~ Z[, 1:2])
##
## Prior probabilities of groups:
##      fake      real
## 0.5373134 0.4626866
##
## Group means:
##      Z[, 1:2]PC1 Z[, 1:2]PC2
## fake   -1.426298  -0.2661719
## real    1.656346   0.3091028
##
## Coefficients of linear discriminants:
##                     LD1
## Z[, 1:2]PC1 1.944574
```

```
## Z[, 1:2]PC2 1.039990
```

```
library(MASS)
par(las = 1)
scatterplot(PC2 ~ PC1 | testing_type, Z, smooth = F, regLine = F, legend = F, cex = 0.85,
            col = c("red", "blue"))
abline(0, -fit$scaling[1] / fit$scaling[2], pch = 5, lwd = 2)
```



The Linear Discriminant Analysis testing set shows that both all real bank notes (red circles) and all fake bank notes (blue triangles) were correctly classified within the plot. The proper classification rate was 100% and there is little chance for a type I or type II error remains low. The training set's model was indeed good to apply to the testing set.

Problem 4

Perform a multidimensional scaling to the USairpollution data set. Identify some outliers and explain how they are different from others.

Load the Data

```r
library(HSAUR2)
```

```
## Loading required package: tools
```

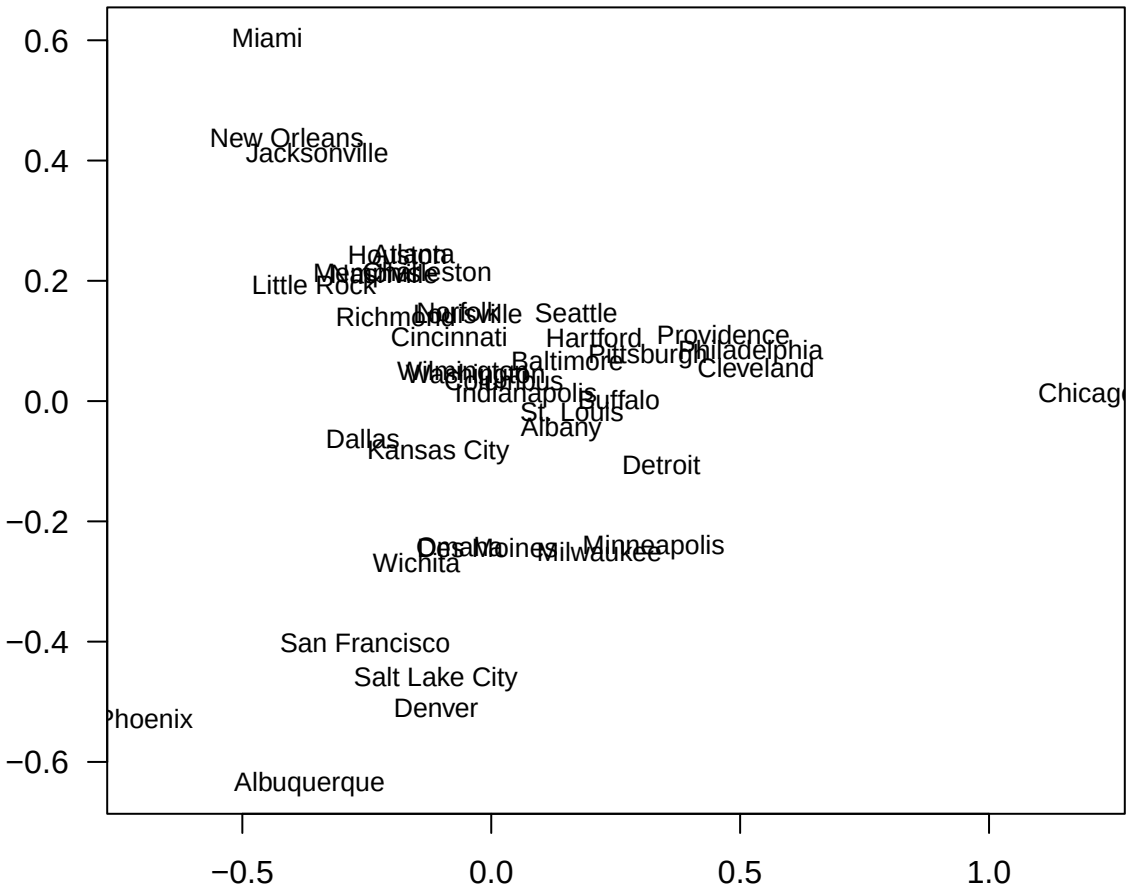
```
##
## Attaching package: 'HSAUR2'
```

```
## The following object is masked from 'package:tidyr':
##
##     household
```

```r
data(USairpollution)
dat <- USairpollution
```

Summarize and Plot the Data

```r
xs <- apply(dat, 2, function(x) (x - min(x)) / (diff(range(x))))
# summary(xs)

# Compute Distance Matrix
poldist <- dist(xs)

# Reduce to 2 Dimensions
pol.mds <- cmdscale(poldist, k = 2, eig = TRUE)

# Reduce to 3 Dimensions
pol.mds3 <- cmdscale(poldist, k = 3, eig = TRUE)

# Plots
par(las = 1, mgp = c(2, 1, 0), mar = c(3, 3, 1, 0.5))
x <- pol.mds$points
plot(x[, 1], x[, 2], type = "n", xlab = "", ylab = "")
text(x[, 1], x[, 2], labels = rownames(x), cex = 0.8)
```



Looking at the plot, the top three outliers appear to be Chicago, Phoenix, and Miami. Chicago, by far, deviates most greatly from the main cluster of cities, followed by Phoenix and Miami. One of the reasons for this finding may pertain to city population. Chicago is one of the top three most populous cities in the United States (and the most populous city in the data set), which means it is likely to have higher pollution levels. The same can be said for Phoenix and Miami, as both cities are also top ten in terms of population in the United States.