# Canonical_Discriminant_Analysis

*Chase Enzweiler*

*10/26/2017*

## Intro to Discriminant Analysis

```
# load data and packages
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.3.2
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 3.3.2
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
library(nnet)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
wine_data <- read.csv("~/Desktop/stat 154/wine.data.csv")
```

### 1.) Sum of Squares Dispersion Function

write a function tss() that computes the total sum of squares of a given variable.

```
# total sum of squares function

TSS <- function(x){
  # x is input vector

  return( sum( (x - mean(x))^2 ) )
}
```

write a function bss() that computes the between groups sum of squares, The below function also calculates the within groups sum of squares when setting WSS = TRUE.

```
# between groups sum of squares can also compute WSS

BSS <- function(x, y, WSS = FALSE){
  # y is vector or factor for response
  # x is vector for the predictor

  # check vectors are same length
  if (length(y) != length(x)){
    stop("vectors must be same length")
  }
```

```r
  # convert y to factor if it is not one
  if (is.factor(y) != FALSE){

    y <- as.factor(y)
  }

  # put the classes in terms of integers where each integer is a different   class factor(class1, class

  y <- as.integer(y)

  # take mean of x

  x_bar <- mean(x)

  # create a matrix with columns y and x

  xy_mat <- cbind(y, x)

  # store the sums

  sums <- 0

  if (WSS == FALSE){
  # computes the formula for each class # BSS
  for (i in unique(y)){

    # observations in class
    n <- length(xy_mat[xy_mat[,1] == i, 2])

    # mean of observations in class
    x_bar_k <- mean(xy_mat[xy_mat[,1] == i, 2])

    sums <- sums + n * (x_bar_k - x_bar)^2

  }

  } else{
    # computes WSS
    for (i in unique(y)){

    # group mean
    x_bar_k <- mean(xy_mat[xy_mat[,1] == i, 2])

    # WSS formula
    sums <- sums + sum( (xy_mat[xy_mat[,1] == i, 2] - x_bar_k)^2)

  }
  }

  return(sums)

}
```

### 2.) Sum of Squares Ratio Functions

use BSS() and TSS() to write a function cor_ratio() that com- putes the correlation ratio eta2 between a variable x and a response y

```
# correlation ratio

cor_ratio <- function(x, y){

  return(BSS(x, y) / TSS(x))

}
```

use bss() and tss() to write a function F_ratio() that computes the F-ratio between a variable x and a response y

```
# F-ratio

F_ratio <- function(x, y){
  # variable x
  # response y

  k <- nlevels(as.factor(y))
  n <- length(x)

  f <- (BSS(x, y)/(k - 1)) / (BSS(x, y, WSS = TRUE)/(n - k))

  return(f)

}
```

### 3.) Discriminant Power of Predictors

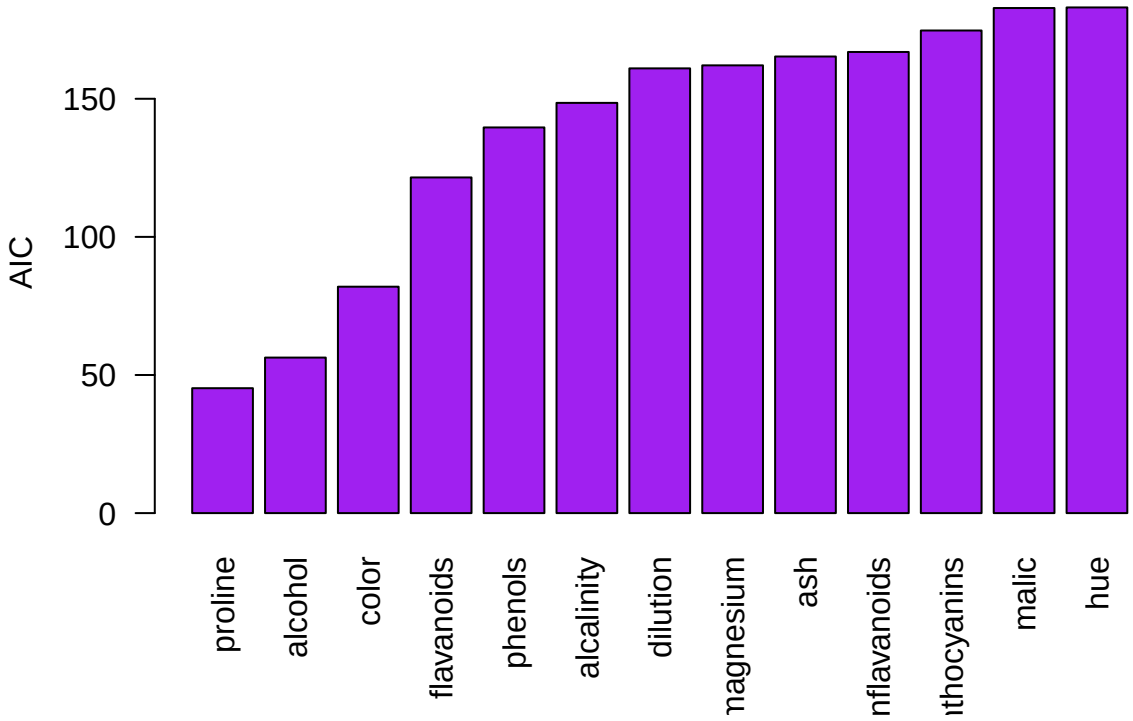
- Run simple logistic regressions for each predictor and the response, and store the values of the AIC statistic.
- Make a table (e.g. data frame) with the predictors ranked by AIC value in increasing order. The smallest the AIC, the more discriminant the predictor.
- Display the AICs in a barchart.

```
# simple logistic regression
# however there are 3 classes so we can not use glm for our logistic regression, we will use multinom f

########################################
# correction update to homework, now run logistic regression on the first two classes
########################################

# create vector to store AIC values
```

```r
AIC <- c()

# vector of predictor names

name_vector <- names(wine_data[,-1])

two_class <- wine_data[1:130,]

two_class$class <- two_class$class - 1

for (i in 1:length(name_vector)){

  # length(name_vector) is 13 because it doesnt include class
  # below wine_data[,i+1] so we dont use class as predictor

  fit <- glm(class ~ two_class[,i+1], data = two_class, family = "binomial")

  AIC[i] <- fit$aic

}
names(AIC) <- name_vector

# sort by increasing AIC
AIC <- sort(AIC)


# data frame of sorted AIC
AIC_table <- as.data.frame(AIC)

AIC_table
```

```
##                       AIC
## proline          45.21948
## alcohol          56.30075
## color            81.96971
## flavanoids      121.51589
## phenols         139.62520
## alcalinity      148.51462
## dilution        161.00793
## magnesium       162.10222
## ash             165.30370
## nonflavanoids   166.94370
## proanthocyanins 174.71983
## malic           182.85454
## hue             183.07125
```

```r
# barplot of sorted AIC
barplot(AIC, las = 2, ylab = "AIC", main = "AIC vs Predictors", col = "purple")
```

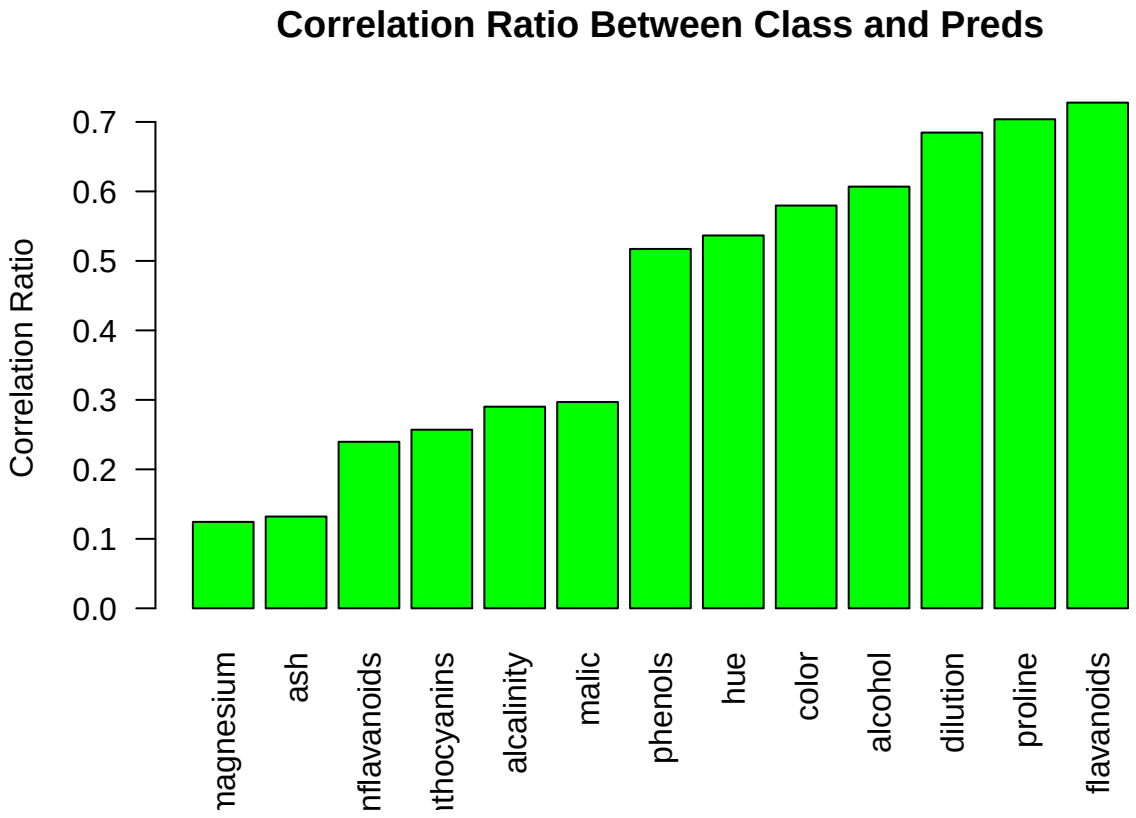



- Calculate correlation ratios for each predictor and the response.
- Make a table (e.g. data frame) with the predictors ranked by n2 value in increasing order. The largest the n2, the more discriminant the predictor.
- Display the n2's in a barchart.

```
# correlations ratios

# store the ratios in this list
c_ratio <- c()


for (i in 1:length(name_vector)){

  # i + 1 column so we dont include class

  c_ratio[i] <- cor_ratio(wine_data[,i+1], wine_data[,1])

}

names(c_ratio) <- name_vector

# sort the ratios
c_ratio <- sort(c_ratio)

# ranked data frame of correlations
as.data.frame(c_ratio)
```

```
##                   c_ratio
## magnesium       0.1243834
## ash             0.1320555
## nonflavanoids   0.2396291
## proanthocyanins 0.2570351
## alcalinity      0.2901855
## malic           0.2968692
## phenols         0.5171961
## hue             0.5365878
## color           0.5796584
## alcohol         0.6068787
## dilution        0.6846532
## proline         0.7038119
## flavanoids      0.7277755
```

```
# barplot of ratios
barplot(c_ratio,las = 2, ylab = "Correlation Ratio", main = "Correlation Ratio Between Class and Preds"
```



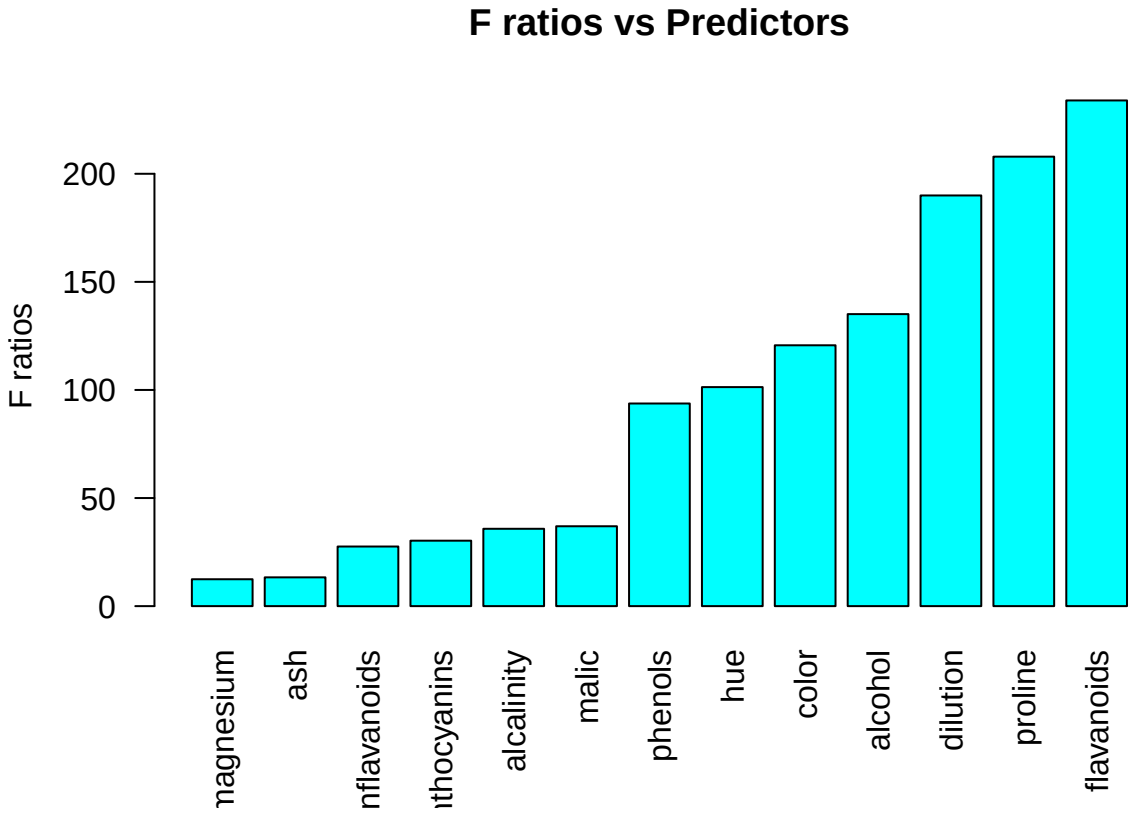

- Calculate F-ratios for each predictor and the response.
- Make a table (e.g. data frame) with the predictors ranked by F-value in increasing order.
- The larger the F, the more discriminant the predictor. Display the F-values in a barchart.

```
# F ratios

# store the F ratios in list
F_list <- c()

for (i in 1:length(name_vector)){
```

```
  # i + 1 column so we dont include class

  F_list[i] <- F_ratio(wine_data[,i+1], wine_data[,1])

}

names(F_list) <- name_vector

# sort by increasing F_list

F_list <- sort(F_list)

as.data.frame(F_list)
```

```
##                    F_list
## magnesium        12.42958
## ash              13.31290
## nonflavanoids    27.57542
## proanthocyanins  30.27138
## alcalinity       35.77164
## malic            36.94342
## phenols          93.73301
## hue             101.31680
## color           120.66402
## alcohol         135.07762
## dilution        189.97232
## proline         207.92037
## flavanoids      233.92587
```

```
# barchart

barplot(F_list, las = 2, ylab = "F ratios", main = "F ratios vs Predictors", col = "cyan")
```




**Variance Functions**

Write a function total_variance() that takes a matrix of predictors, and returns the (sample) variance-covariance matrix V. Do NOT use var() to create total_variance().

```
# function to create sample variance covariance matrix
total_variance <- function(X){
  # X is matrix of predictors

  # number of observations

  n <- dim(X)[1]

  # mean center but dont scale

  X <- scale(X, center = TRUE, scale = FALSE)

  return( (t(X) %*% X) / (n-1) )

}

total_variance(iris[,1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
```

```
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

Write a function between_variance() that takes a matrix of predictors, and a response vector (or factor), and returns the (sample) Between-variance matrix B. Do NOT use var() to create between_variance().

```r
# function for between variances

between_variance <- function(X, y){
  # X is matrix of predictors
  # y is response variable

  # number of total observations
  n <- dim(X)[1]

  # convert y to classes referring to integers
  y <- as.integer(as.factor(y))

  # column binded matrix of x and y
  xy_mat <- cbind(y, X)

  # matrix to store sums of other matrices
  # subtract one because we dont want to consider response
  sum_mat <- 0

  # centroid g of predictors

  g <- colMeans(xy_mat[,-1])

  # variances for each class k
  for (i in unique(y)){

    # filter the xy matrix by class to get a group
    group_mat <- xy_mat[xy_mat[,1] == i, -1]

    # observations in group
    n_k <- dim(group_mat)[1]

    # centroids of group

    g_k <- colMeans(group_mat)

    sum_mat <- sum_mat + (n_k / (n-1)) * (g_k - g) %*% t(g_k - g)


  }

  rownames(sum_mat) <- colnames(sum_mat)

  return(sum_mat)

}
```

```r
between_variance(iris[,1:4], iris$Species)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.4242425 -0.13391051    1.1090497   0.4783848
## Sepal.Width    -0.1339105  0.07614049   -0.3841584  -0.1539105
## Petal.Length    1.1090497 -0.38415839    2.9335758   1.2535168
## Petal.Width     0.4783848 -0.15391051    1.2535168   0.5396868
```

Write a function within_variance() that takes a matrix of predictors, and a response vector (or factor), and returns the (sample) Within-variance matrix W. Do NOT use var() to create within_variance().

```r
# within group variance
within_variance <- function(X, y){
  # X is matrix of predictors
  # y is response variable

  n <- dim(X)[1]

  # converts y to integer classes
  y <- as.integer(as.factor(y))

  xy_mat <- cbind(y, X)

  sum <- 0

  # loop through each group
  for (i in unique(y)){

    X_k <- as.matrix(xy_mat[xy_mat[,1] == i, -1])

    n_k <- dim(X_k)[1]

    W_k <- total_variance(X_k)

    sum <- sum + ((n_k - 1)/(n - 1)) * W_k
  }

  return(sum)
}

within_variance(iris[,1:4], iris$Species)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.26145101  0.09147651   0.16526577  0.03788591
## Sepal.Width    0.09147651  0.11383893   0.05450201  0.03227114
## Petal.Length   0.16526577  0.05450201   0.18270201  0.04209262
## Petal.Width    0.03788591  0.03227114   0.04209262  0.04131946
```

Now confiirm that the sum of the within group and between group variances equal the total variance.

```r
# test our functions using iris data set

total_variance(iris[,1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

```
between_variance(iris[,1:4], iris$Species) + within_variance(iris[,1:4], iris$Species)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

### Canonical Discriminant Analysis

Use the predictors and response of the wine data, to write code in R that allows you to find the eigenvectors $u_k$.

```
# find eigenvectors u_k

# within variance matrix W
W <- within_variance(wine_data[,-1], wine_data[,1])

# between variance matrix B
B <- between_variance(wine_data[,-1], wine_data[,1])

# decompose B into B = C %*% t(C)

# number of total observations in wine data
n <- dim(wine_data)[1]

# col means of predictors
xj_bar <- colMeans(wine_data[,-1])

# col means of predictors for each class
xjk_bar <- list()

# number of observations in each wine class

n_k <- c()

for (i in unique(wine_data$class)){

  xjk_bar[[i]] <- colMeans(wine_data[wine_data[,1] == i, -1])

  n_k[i] <- dim(wine_data[wine_data[,1] == i, -1])[1]

}

# create matrix C
```

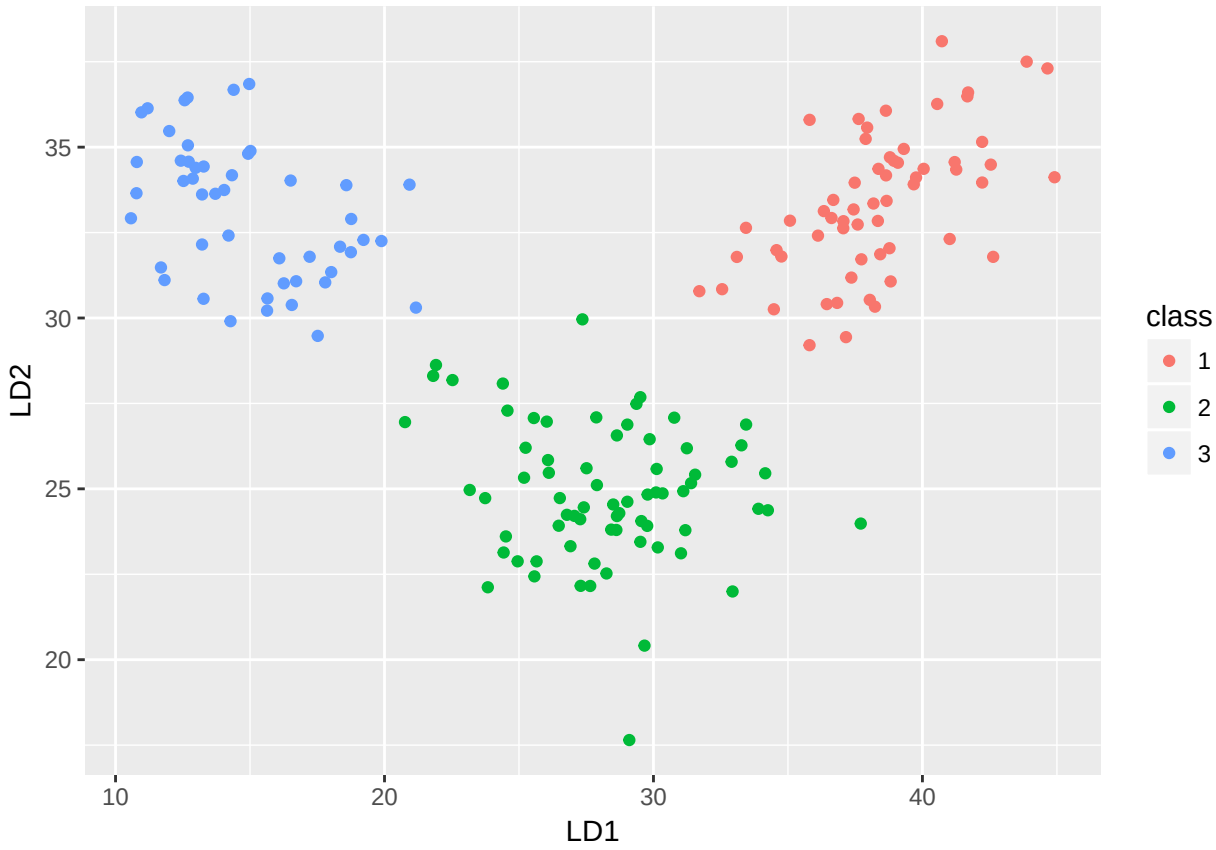
```r
C <- sqrt(n_k[1]/ (n-1)) * (xjk_bar[[1]] - xj_bar)

for (i in 2:length(unique(wine_data$class))){

  new_row <- sqrt(n_k[i]/ (n-1)) * (xjk_bar[[i]] - xj_bar)

   C <- cbind(C, new_row)
}

colnames(C) <- c("class 1", "class 2", "class 3")


# now we can use eigen value decomposition to find the eigenvectors w of t(C) %*% solve(W) %*% C

w <- eigen(t(C) %*% solve(W) %*% C)

# now recover the eigenvectors u with w, u = solve(W) %*% C %*% w

u <- solve(W) %*% C %*% w$vectors

# These are our eigenvectors u
u
```

```
##                         [,1]          [,2]          [,3]
## alcohol          1.222609554  1.7814577940  9.714451e-17
## malic           -0.500847689  0.6240254979 -1.776357e-15
## ash              1.118580020  4.7936058021 -1.421085e-14
## alcalinity      -0.469155877 -0.2991204731  8.881784e-16
## magnesium        0.006557047 -0.0009456156  1.387779e-17
## phenols         -1.873169990 -0.0658249947  0.000000e+00
## flavanoids       5.034678679 -1.0053690476  3.996803e-15
## nonflavanoids    4.533472756 -3.3327580255  1.199041e-14
## proanthocyanins -0.406403118 -0.6275153789  2.033096e-15
## color           -1.076090082  0.5174619938 -1.998401e-15
## hue              2.479274325 -3.0971098347  1.332268e-15
## dilution         3.508289345  0.1045914210 -2.442491e-15
## proline          0.008156412  0.0058299061 -8.239937e-18
```

```r
# the u are the vectors associated with the canonical axes and we keep the minimum of k-1 and P, therfo
```

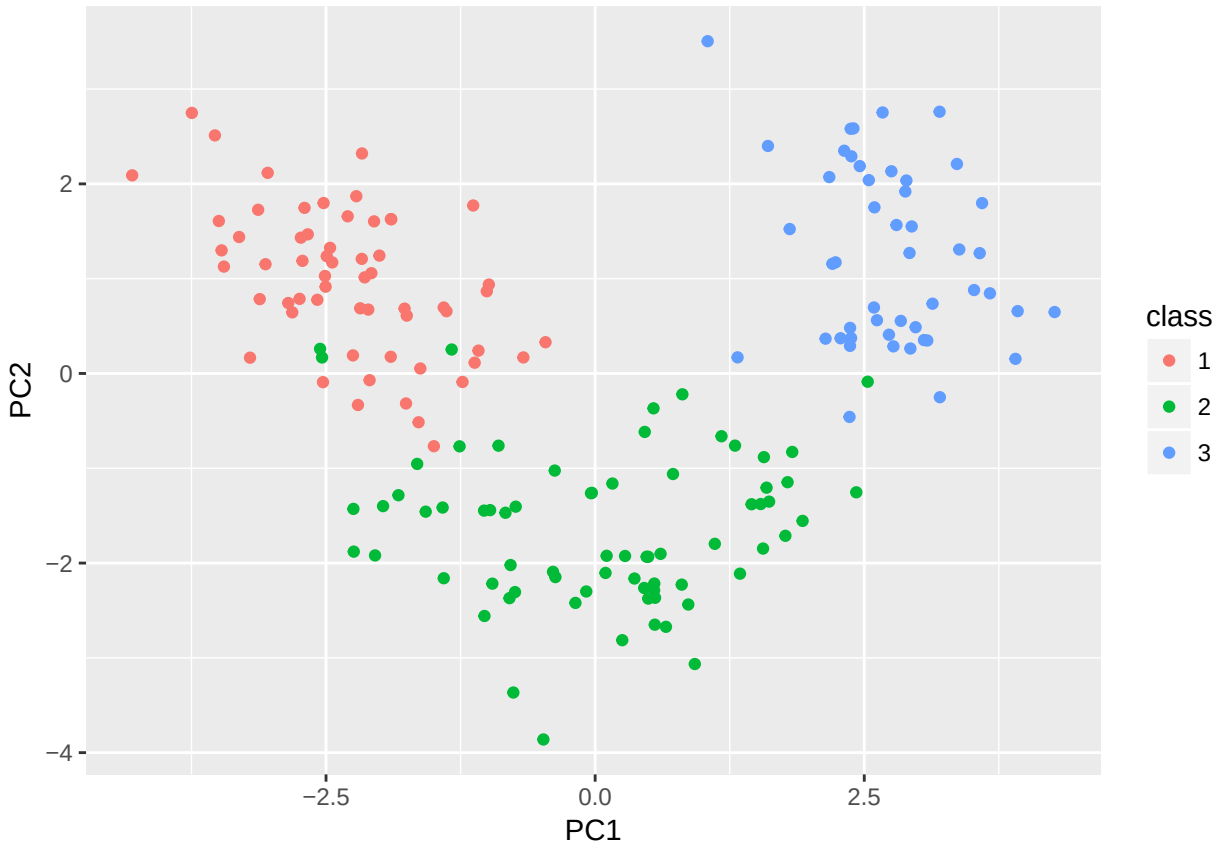
Obtain the linear combinations $z_k$ and make a scatterplot of the wines. Add color to the dots indicating the different classes.

```r
# obtain the linear combination z_k
z_k <- as.matrix(wine_data[,-1]) %*% u[,1:2]

# create a scatter plot

wine_lda <- data.frame(z_k)

colnames(wine_lda) <- c("LD1", "LD2")

wine_lda$class <- factor(wine_data$class)

ggplot(data = wine_lda, aes(LD1, LD2, color = class)) + geom_point()
```




Obtain a scatterplot of the wines but this time using the first two principal components on the standardized predictors. Add color to the dots indicating the different classes. How does this compare to the previous scatterplot?

```
# find the first two principal components
pca <- prcomp(wine_data[,-1], scale = TRUE)

# the principal components are contained in pca$rotation
# need to use on standardized data

wine_pca <- as.matrix(scale(wine_data[,-1])) %*% pca$rotation[,1:2]

wine_pca <- data.frame(wine_pca)

wine_pca$class <- factor(wine_data$class)

ggplot(data = wine_pca, aes(PC1, PC2, color = class)) + geom_point()
```




Comparing the scatter plot obtained from linear combinations $z_k$ and the scatter plot made from the two PC's, we notice that the plots are mirrored where class 1 and class 3 are on the right and left side respectively in the first scatter plot and this is the opposite in the principal component scatter plot. In the PC plot the groups are not as well seperated and the within group points are more spread out as well.

Calculate the correlations between zk and the predictors. How do you interpret each score?

```
# calculate the correlations between z_k and the predictors
cor(z_k[,1], wine_data[,-1])
```

```
##        alcohol     malic        ash alcalinity magnesium   phenols
## [1,] 0.2798969 -0.489176 0.01918243 -0.5299978 0.1935927 0.7548212
##      flavanoids nonflavanoids proanthocyanins      color       hue
## [1,]  0.8984936    -0.5152212       0.5320387 -0.3441133 0.6840759
##       dilution   proline
## [1,] 0.8503779 0.6148947
```

```
cor(z_k[,2], wine_data[,-1])
```

```
##       alcohol     malic       ash alcalinity magnesium    phenols
## [1,] 0.816218 0.3178155 0.4045125 -0.2148215 0.3355196 0.07008972
##       flavanoids nonflavanoids proanthocyanins     color        hue
## [1,] -0.02635971   -0.02507846     -0.05042644 0.7665231 -0.3780354
##        dilution   proline
## [1,] -0.2031988 0.6717132
```

We can interpret these scores as the predictors representation on the canonical axes, we see that dilution and flavornoids are highly correlated with the first canonical axes therefore they are well represented. Also see that alcohol is highly correlated with the second canonical axis.

Create a matrix of size n × K, with the squared Mahalanobis distances d2(xi, gk) of each observation xi (i.e. each wine) to the each of the k centroids gk. Finally, assign each observation to the class Gk for which the Mahalanobis distance d2(xi,gk) is the smallest. And create a confussion matrix comparing the actual class versus the predicted class

```
# n x k matrix of the squared mahalanobis distances

# centroids of each predictor with respect to the class g_k were calculated above as xjk_bar

# matrix of the mahalanobis distance

D2 <- matrix(0,nrow = dim(wine_data)[1] ,ncol = length(unique(wine_data$class)))

for (k in unique(wine_data$class)){

  for (i in 1:dim(wine_data)[1]){

    D2[i,k] <- (as.matrix(wine_data[i,-1]) - xjk_bar[[k]]) %*% solve(W) %*% t(as.matrix(wine_data[i,-1]))
  }
}

# use which.min on each row of matrix, will give col with smallest distance, column corresponds the cla

# predicted class for each observation according to mahalanobis distance
predicted_class <- c()

for (i in 1:dim(wine_data)[1]){

  predicted_class[i] <- which.min(D2[i,])

}

# check to see how many observations were predicted correctly
wine_data$class == predicted_class
```

```
##   [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [15] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [29] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [43] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [57] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [71] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [85] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [99] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [113] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [127] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [141] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [155] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [169] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
# all the observations were predicted into the true classes

# confusion matrix

confusion_mat <- diag(c(length(which(wine_data$class ==1)), length(which(wine_data$class ==2)), length(
```

```r
confusion_mat <- rbind(confusion_mat ,c(59, 71, 48))

confusion_mat <- cbind(confusion_mat ,c(59, 71, 48,(59+71+48)))

colnames(confusion_mat) <- c("True 1", "True 2", "True 3", "total")
rownames(confusion_mat) <- c("pred 1", "pred 2", "pred 3", "total")

confusion_mat <- data.frame(confusion_mat)

confusion_mat
```

```
##        True.1 True.2 True.3 total
## pred 1     59      0      0    59
## pred 2      0     71      0    71
## pred 3      0      0     48    48
## total      59     71     48   178
```