Lab 10: Canonical Discriminant Analysis

Stat 154, Spring 2018

Introduction

In this lab you will be working with the `iris` data set. The main purpose is to work around the concepts of how total dispersion (i.e. total sum of squares) can be broken down into between-groups and within-groups dispersion. These concepts are the root of linear discriminant analysis and quadratic discriminant analysis (to be reviewed in HW5).

1) Sum-of-Squares Dispersion Functions

Consider a variable X and a categorical variable Y. Assume that Y represents the class (or group) of each observation. Let k be an index for the classes: $k = 1, \dots, K$; each class is of size n_k.

Function `tss()`: write a function `tss()` that computes the total sum of squares of a given variable:

$$\text{tss} = \sum_{i=1}^{n} (x - \bar{x})^2$$

The function `tss()` should take one argument `x`, the input vector.

Here's how you should be able to call `tss()`

```
tss(iris$Sepal.Length)
```

```
## [1] 102.1683
```

Function `bss()`: write a function `bss()` that computes the between groups sum of squares:

$$\text{bss} = \sum_{k=1}^{K} n_k (\bar{x}_k - \bar{x})^2$$

where:

- n_k is the number of individuals in class k
- $\bar{x}_k$ is the average of class k

The function `bss()` takes two arguments:

- `x` = vector for the predictor variable

- `y` = vector (or factor) for the response variable

Here's how you should be able to call `bss()`

```
bss(iris$Sepal.Length, iris$Species)
```

```
## [1] 63.21213
```

Function `wss()`: write a function `wss()` that computes the wetween groups sum of squares:

$$\text{wss} = \sum_{k=1}^{K} \sum_{i \in G_k} (x_{ik} - \bar{x}_k)^2$$

where:

- x_{ik} is the i-th individual in class k
- $\bar{x}_k$ is the average of group k
- G_k represents the group of individuals in class k

The function `wss()` takes two arguments:

- `x` = vector for the predictor variable
- `y` = vector (or factor) for the response variable

Here's how you should be able to call `wss()`

```
wss(iris$Sepal.Length, iris$Species)
```

```
## [1] 38.9562
```

2) Sum-of-Squares Ratio Functions

Function `cor_ratio()`: use `bss()` and `tss()` to write a function `cor_ratio()` that computes the correlation ratio eta^2 between a variable `x` and a response `y`.

$$\eta^2(x, y) = \frac{\text{bss}}{\text{tss}}$$

Here's how you should be able to call `cor_ratio()`

```
cor_ratio(iris$Sepal.Length, iris$Species)
```

```
## [1] 0.6187057
```

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

$$F = \frac{\text{bss}/(K-1)}{\text{wss}/(n-K)}$$

Here's how you should be able to call `F_ratio()`

```r
F_ratio(iris$Sepal.Length, iris$Species)
```

```
## [1] 119.2645
```

---

3) Discriminant Power of Predictors

For this part of the lab, you will rank the predictors (e.g. `Sepal.Length`, `Sepal.Width`, `Petal.Length`, and `Petal.Width`) using three approaches: 1) simple logistic regressions, 2) correlation ratios, and 3) F-ratios.

The first approach consists of running simple logistic regressions:

- 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.

The second approach consists of computing correlation ratios:

- Calculate correlation ratios for each predictor and the response.
- Make a table (e.g. data frame) with the predictors ranked by η^2 value in increasing order. The largest the η^2, the more discriminant the predictor.
- Display the η^2's in a barchart.

The third approach consists of computing F-ratios:

- 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 largest the F, the more discriminant the predictor.
- Display the F-values in a barchart.

How do the AIC values from the logistic regression compare to the sum of squares ratios?

---

4) Variance functions

Function `total_variance()`: Write a function `total_variance()` that takes a matrix of predictors, and returns the (sample) variance-covariance matrix $\mathbf{V}$. Do NOT use `var()` to create `total_variance()`.

Here's how yo should be able to invoke `total_variance()`, and compare it with the `var()` function.

```
# test total_variance()
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
```

```
# compare with var()
var(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
```

Function `between_variance()`: 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()`.

Here's how yo should be able to invoke `between_variance()` on `iris` data

```
# test between_variance()
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
```

Function `within_variance()`: 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()`.

Here's how yo should be able to invoke `within_variance()` on `iris` data

```
# test within_variance()
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
```

Confirm that $\mathbf{V} = \mathbf{B} + \mathbf{W}$

```
# confirm V = B + W
Viris <- total_variance(iris[ ,1:4])
Viris
```

```
##              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
```

```
# B + W
Biris <- between_variance(iris[ ,1:4], iris$Species)
Wiris <- within_variance(iris[ ,1:4], iris$Species)
Biris + Wiris
```

```
##              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
```

Challenge

Use the predictors and response of the `iris` data, to write code in R that allows you to find the eigenvectors $\mathbf{u_k}$.

```
# predictors, response, and indices
X <- as.matrix(iris[ ,-5])
y <- as.factor(iris$Species)
k <- nlevels(y)
n <- nrow(X)
nk <- as.vector(table(y))

# matrix of group means (i.e. centroids)
centroids <- matrix(0, ncol(X), k)
for (j in 1:ncol(X)) {
  centroids[j,] <- tapply(X[,j], y, FUN=mean)
}
dimnames(centroids) <- list(colnames(X), levels(y))
centroids
```

```
##              setosa versicolor virginica
## Sepal.Length  5.006      5.936     6.588
## Sepal.Width   3.428      2.770     2.974
## Petal.Length  1.462      4.260     5.552
## Petal.Width   0.246      1.326     2.026
```

```
# decomposing between-class matrix:  B = CC'
gm <- colMeans(X)
centroids_centered <- sweep(centroids, 1, gm, FUN="-")
C <- sweep(centroids_centered, 2, sqrt(nk/n), FUN="*")
C
```

```
##                  setosa  versicolor   virginica
## Sepal.Length -0.4834346  0.05350112  0.42993350
## Sepal.Width   0.2140045 -0.16589198 -0.04811252
## Petal.Length -1.3255962  0.28982984  1.03576638
## Petal.Width  -0.5504073  0.07313103  0.47727622
```

```
# within-groups covariance matrix
W <- within_variance(X, y)
```

Thus, we can diagonalize (i.e. EVD) the following symmetric matrix:

$$\mathbf{C}^\mathsf{T}\mathbf{W}^{-1}\mathbf{C}$$

and then use the eigenvector $\mathbf{w}$ to recover $\mathbf{u}$

$$\mathbf{u} = \mathbf{W}^{-1}\mathbf{C}\mathbf{w}$$

```
# eigen-decomposition
EIG <- eigen(t(C) %*% solve(W) %*% C)
lam <- EIG$values[1:(k - 1)]
U <- solve(W) %*% C %*% EIG$vectors[,1:(k-1)]

head(U)
```

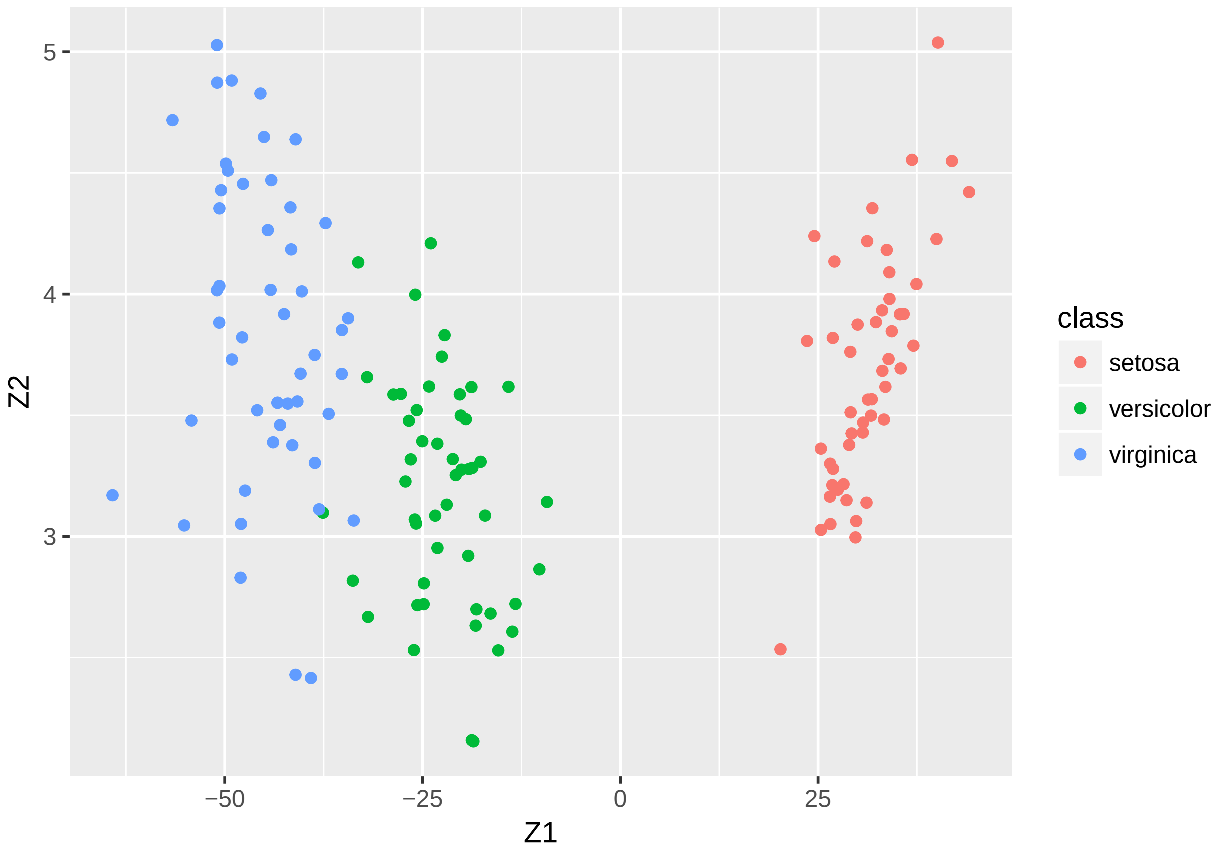
```
##                   [,1]        [,2]
## Sepal.Length   4.721802  0.01291986
## Sepal.Width    8.736043  1.16028317
## Petal.Length -12.531910 -0.49955273
## Petal.Width  -16.000477  1.52193558
```

Obtain the linear combinations $\mathbf{z_k}$ and make a scatterplot of the wines. Add color to the dots indicating the different classes. (10 pts)

It is possible that the scale of your scatterplot is different, or even that the shape is different (e.g. you may have an inverted image of my plot). The important thing is the relative position of the cloud of points.

```
# canonical scores (components)
Z <- as.data.frame(X %*% U)
names(Z) <- c("Z1", "Z2")
Z$class <- y

# scatterplot (canonical variables)
ggplot(data = Z, aes(x = Z1, y = Z2, color = class)) +
  geom_point()
```

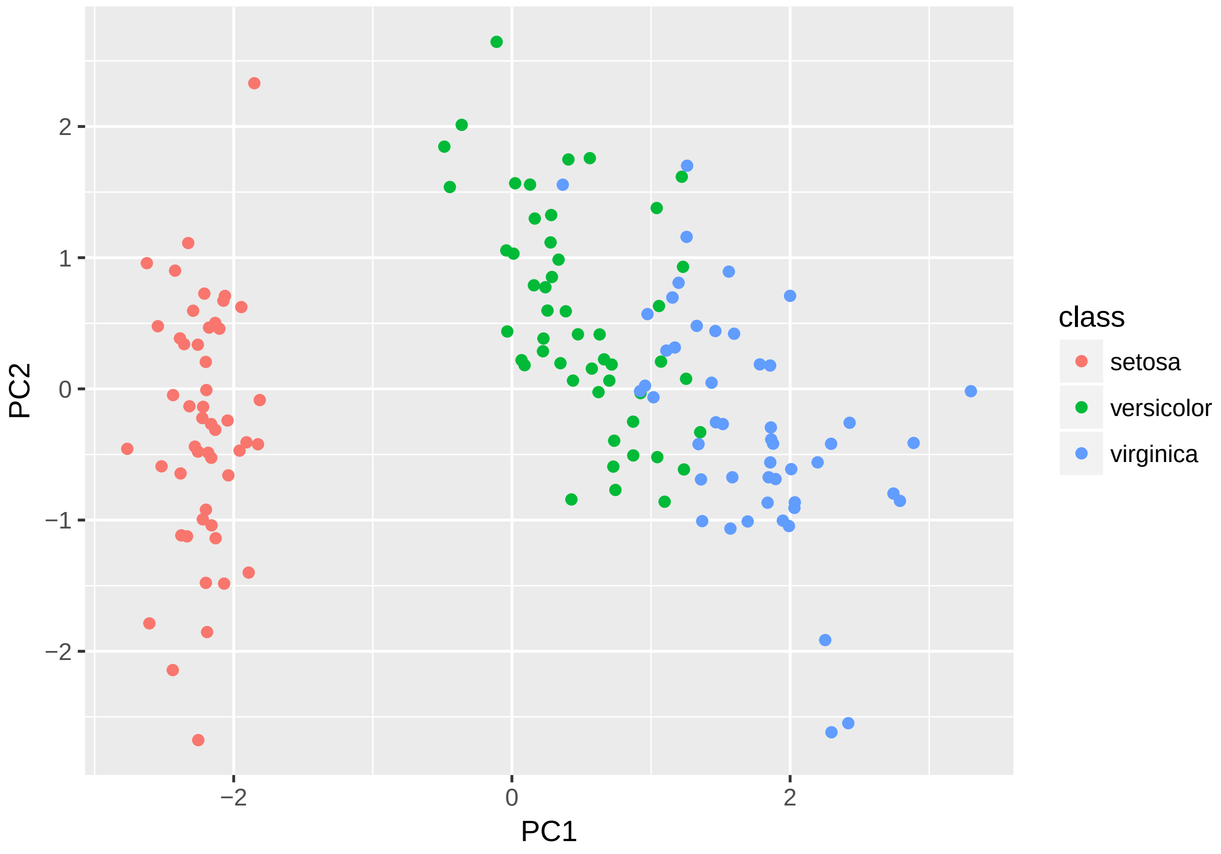


Obtain a scatterplot of the wines but this time using the first two principal components on the standardized predictors. Add color to the dots indicating the different classes. How does this compare to the previous scatterplot? (10 pts)

```
# principal components analysis (PCA)
pca <- prcomp(X, scale. = TRUE)
PC <- as.data.frame(pca$x)
PC$class <- y

# PCA scatterplot
```

```
ggplot(data = PC, aes(x = PC1, y = PC2, color = class)) +
  geom_point()
```


PC2
PC1
class
setosa
versicolor
virginica