STAT_8320_HW6

JAYADITYA NATH

2024-05-04

```
# Loading the required libraries

library(MASS)
library(class)
library(klaR)
library(CovTools)
library(MVN)
library(ggplot2)
library(psych)
library(dplyr)
```

PROBLEM 1.

Considering the matrix :

$$X = \begin{bmatrix} 7 & 4 & 3 \\ 4 & 1 & 7 \\ 6 & 3 & 5 \\ 7 & 6 & 1 \\ 8 & 5 & 6 \\ 7 & 3 & 9 \\ 5 & 3 & 3 \\ 9 & 2 & 8 \\ 7 & 4 & 5 \\ 8 & 2 & 2 \end{bmatrix}$$

Part (a)

```
## [1] "The sample covariance matrix is : "
```

```
##           [,1]       [,2]      [,3]
## [1,] 2.1777778  0.6222222  0.200000
## [2,] 0.6222222  2.2333333 -1.744444
## [3,] 0.2000000 -1.7444444  6.988889
```

Part (b)

```
## [1] "The total variance is : 11.4"
```

Part (c)

```
## [1] "The eigen values are : "
```

```
##          [,1]     [,2]     [,3]
## [1,] 7.560167 0.000000 0.000000
## [2,] 0.000000 2.622511 0.000000
## [3,] 0.000000 0.000000 1.217323
```

```
## [1] "The eigen vectors are : "
```

```
##               [,1]      [,2]       [,3]
## [1,] -0.0006741816 0.8267432  0.5625791
## [2,] -0.3112901415 0.5344539 -0.7857846
## [3,]  0.9503146812 0.1756551 -0.2569967
```

Part (d)

```
## [1] "The proportion of variation accounted for by the first principal component is : 0.6631725019118
```

Part (e)

```
## [1] "The cumulative variation accounted for by the first two principal components is : 0.893217318895
```

Part (f)

```
## [1] "The first principal component is : "
```

```
##             [,1]
##  [1,]  1.6010642
##  [2,]  6.3382159
##  [3,]  3.8136579
##  [4,] -0.9221454
##  [5,]  4.1400439
##  [6,]  7.6142424
##  [7,]  1.9137027
##  [8,]  6.9738695
##  [9,]  3.5016936
## [10,]  1.2726556
```

```
## [1] "The second principal component is : "
```

```
##            [,1]
##  [1,]  8.451983
##  [2,]  5.071012
##  [3,]  7.442096
##  [4,]  9.169581
##  [5,] 10.340146
##  [6,]  8.971460
##  [7,]  6.264043
##  [8,]  9.914837
##  [9,]  8.803293
## [10,]  8.034164
```

Part (g)

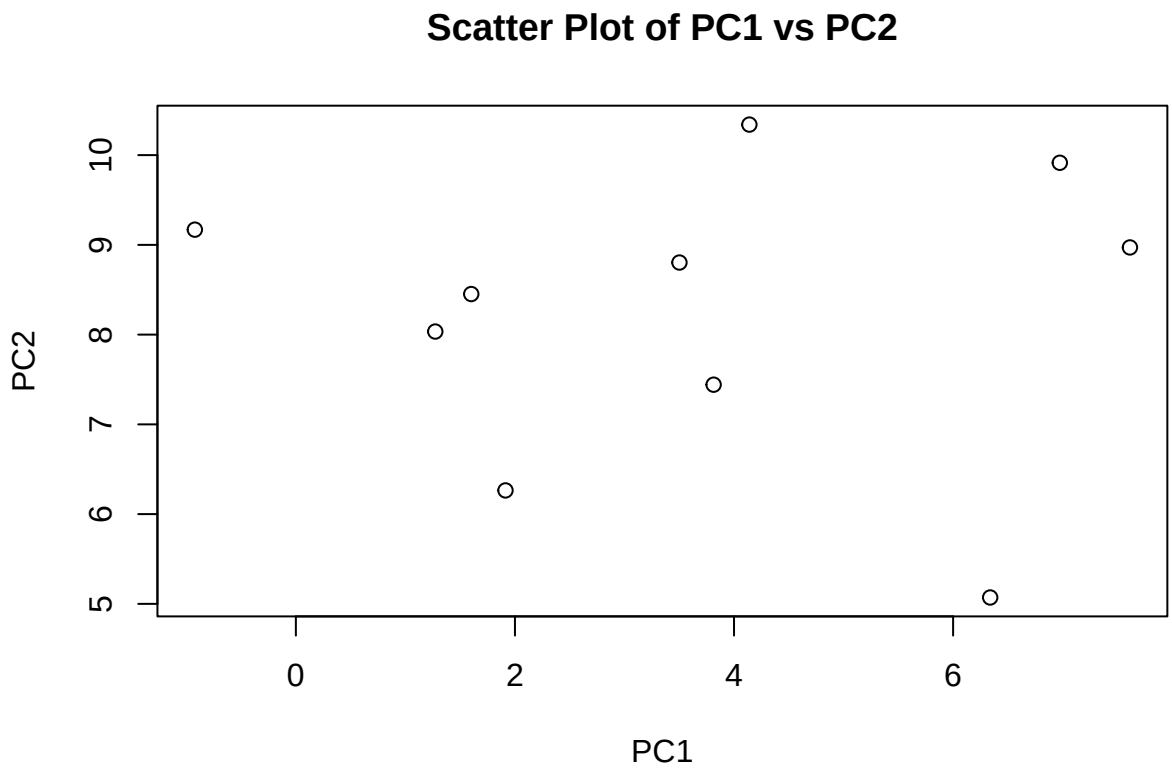
```
## [1] "Loading Coefficients for PC1 and PC2:"
```

```
##               [,1]       [,2]
## [1,] -0.0005490225 0.39653074
## [2,] -0.2535003842 0.25634005
## [3,]  0.7738925996 0.08424942
```

For the first two variables, it seems that both of them load moderately on the second factor, but, for the first factor the results are completely contrasted.

For the third variable, there seems to be a strong to moderate loading when a transition is made from the first factor to the second one.

Part (h)



Comment : It does not seem that any of the principal components seem to affect any of the three traits strongly.

Part (i)

```
##            [,1]       [,2]        [,3]
## [1,] 1.00000000  0.2821382  0.05126479
## [2,] 0.28213825  1.0000000 -0.44154635
## [3,] 0.05126479 -0.4415464  1.00000000
```

Part (j)

```
## [1] "The eigen values are : "

##          [,1]     [,2]      [,3]
## [1,] 1.501733 0.000000 0.0000000
## [2,] 0.000000 1.046441 0.0000000
## [3,] 0.000000 0.000000 0.4518267

## [1] "The eigen vectors are : "

##            [,1]       [,2]       [,3]
## [1,] -0.3446268 0.84391391  0.4111468
## [2,] -0.7218936 0.04172653 -0.6907449
## [3,]  0.6000849 0.53485345 -0.5948360

## [1] "The proportion of variation accounted for by the first principal component is : 0.5005775985491

## [1] "The proportion of variation accounted for by the second principal component is : 0.348813505078
```

PROBLEM 2.

```
# Loading the data
data_2 = read.table("C:/Users/Jayaditya Nath/Documents/decathlon.dat")
data_2[, c(1, 5, 6, 10)] = -data_2[, c(1, 5, 6, 10)]
data_2_cov = data_2[, -11]
```

Part (a)

As Principal Component Analysis is not invariant to change in scale of the variables and here, we are rescaling some of the variables, it makes sense to use the correlation matrix for better interpretability.

Part (b)

```
pca_mod = prcomp(data_2_cov,scale. = T)

summary(pca_mod)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.2413 1.4422 0.85759 0.82809 0.61340 0.54964 0.53433
## Proportion of Variance 0.5023 0.2080 0.07355 0.06857 0.03763 0.03021 0.02855
## Cumulative Proportion  0.5023 0.7103 0.78389 0.85246 0.89009 0.92030 0.94885
##                           PC8     PC9   PC10
## Standard deviation     0.47305 0.45248 0.2881
## Proportion of Variance 0.02238 0.02047 0.0083
## Cumulative Proportion  0.97123 0.99170 1.0000
```

It seems that the first PC accounts for 50% of the variability and the second one accounts for 21% of the variability.

Part (c)

Yes, I would definitely use the first 2 PC's in the analysis because most of the variation(roughly 70%) in the data is explained by the two of these components.

Part (d)

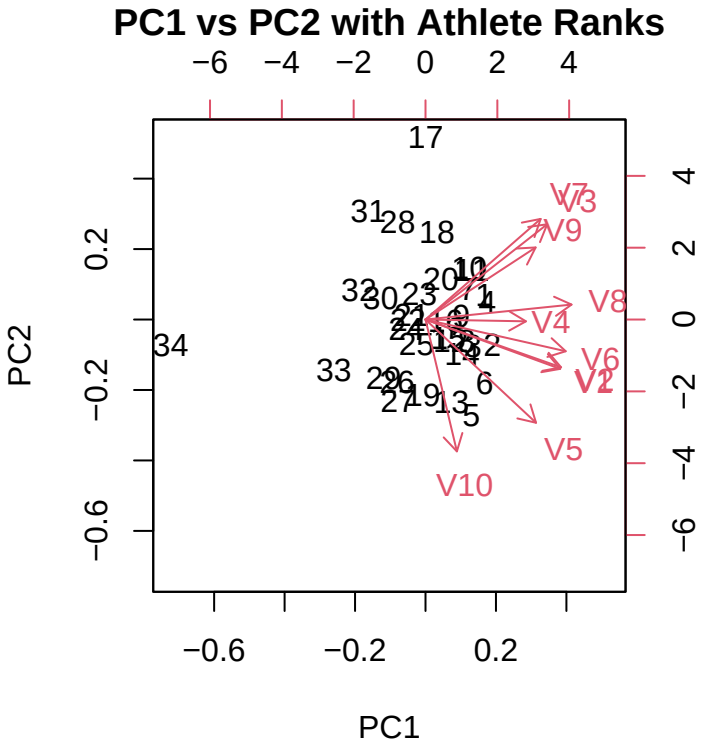
```r
pca_mod$rotation[,1:2]
```

```
##             PC1          PC2
## V1  0.35863450 -0.204157325
## V2  0.36119457 -0.197895437
## V3  0.32399083  0.394274668
## V4  0.26747658 -0.007412489
## V5  0.29436691 -0.427197688
## V6  0.37338818 -0.131247472
## V7  0.30669150  0.416387386
## V8  0.38894727  0.061929264
## V9  0.29327539  0.298380393
## V10 0.08357796 -0.545601880
```

From the loadings of the first PC, it seems to influence each variable almost similarly and thus can be interpreted as a measure of the total overall score of an athlete. However, the loadings of the second PC seem to be contrasted for several variables such as 100 m race, long jump, etc. with variables such as discus, shot put, etc. and thus there is no clear interpretation for this component.

Part (e)

```r
# Extract PC scores
biplot(pca_mod,main="PC1 vs PC2 with Athlete Ranks")
```



We can observe from the plot that most of the variables have been combined and transformed into

PC1(X-axis of the plot) with the highest values of ranks of the athletes towards the lowest values of PC1 and the converse is also true.

Part (f)

```
# Extract PC scores
pc_scores = pca_mod$x

# Create a dataframe with PC scores and athlete ranks
pc_data = data.frame(PC1 = pc_scores[,1], PC2 = pc_scores[,2], Rank = rank(data_2$V11))

# Correlate the first PC score with the overall decathlon point score
cor_pc1 = cor(pc_data$PC1, data_2$V11)

# Correlate the second PC score with the overall decathlon point score
cor_pc2 = cor(pc_data$PC2, data_2$V11)

# Print correlation results
print(paste("Correlation between PC1 and overall decathlon point score:", cor_pc1))
```

```
## [1] "Correlation between PC1 and overall decathlon point score: 0.99146188749459"
```

```
print(paste("Correlation between PC2 and overall decathlon point score:", cor_pc2))
```

```
## [1] "Correlation between PC2 and overall decathlon point score: -0.0169026558006945"
```

The correlation coefficient between the first and second principal components and the overall decathlon point scores indicates the strength and direction of the linear relationship between them. A positive correlation suggests that higher values of PC1 and PC2 are associated with higher decathlon point scores, while a negative correlation suggests the opposite. The results here suggest that the PC1 can be considered as a measure of the overall decathlon score, where as the interpretation of PC2 as a measure of the overall score is not possible.

PROBLEM 3.

Part (a)

FALSE

Part (b)

$\frac{6.338464}{13} \times 100\ \% = \mathbf{48.7574}\ \%$

Part (c)

TRUE

Part (d)

FALSE

Part (e)

TRUE

Part (f)

0.6497988315 for Initial Factor Analysis

0.6498017064 for Varimax Rotation

Part (g)

TRUE

Part (h)

L

PROBLEM 4.

```
# Loading the data
data_4 = read.table("C:/Users/Jayaditya Nath/Documents/dataset_factor.csv",sep = ",",header = T)
```

Part (a)

```
fa_mod_1 = fa(data_4,rotate="none",fm="mle")
fa_mod_1$PVAL
```

```
## [1] 6.109624e-69
```

```
fa_mod_2 = fa(data_4,nfactors=2,rotate="none",fm="mle")
fa_mod_2$PVAL
```

```
## [1] 0.5676271
```

```
fa_mod_3 = fa(data_4,nfactors=3,rotate="none",fm="mle")
fa_mod_3$PVAL
```

```
## [1] NA
```

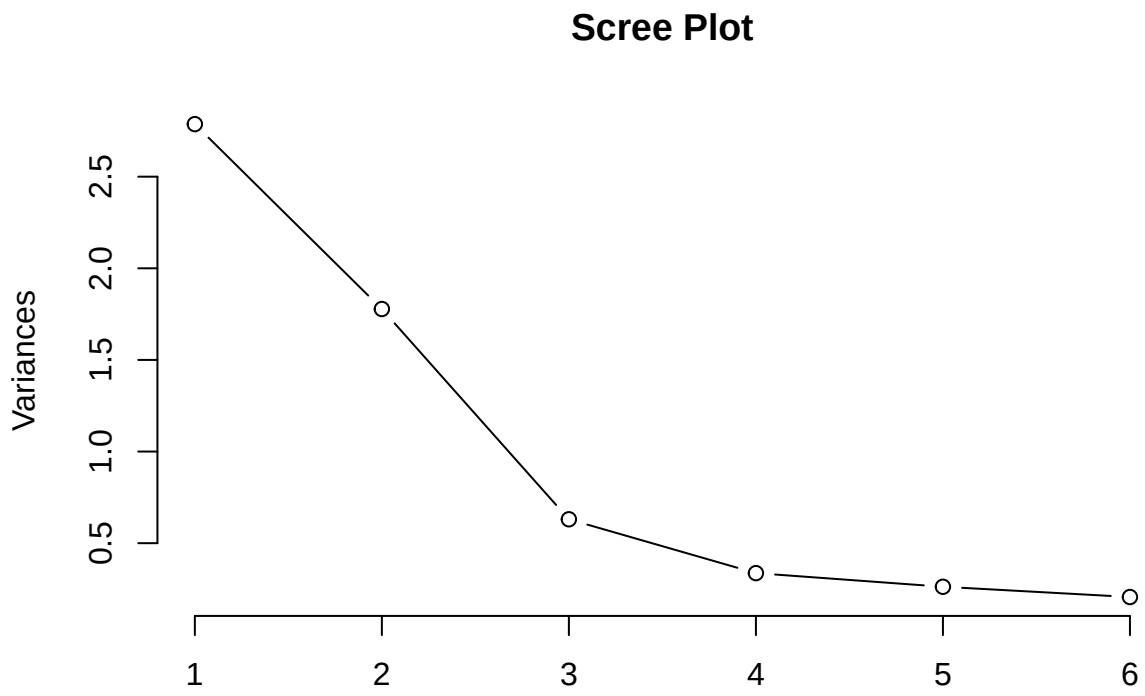
It is clear from the results that 2 factors should be considered here.

Part (b)

```
pca_mod = principal(data_4,nfactors = 2,rotate = "none")
pca_mod$loadings
```

```
##
## Loadings:
##      PC1    PC2
## BIO   0.782 -0.454
## GEO   0.771 -0.416
## CHEM  0.748 -0.513
## ALG   0.547  0.701
## CALC  0.635  0.664
## STAT  0.564  0.451
##
##                  PC1   PC2
## SS loadings    2.787 1.778
## Proportion Var 0.464 0.296
## Cumulative Var 0.464 0.761
```

```
screeplot(prcomp(data_4,scale=T),type="lines",main = "Scree Plot")
```



The screeplot shows that most of the variability has been accounted for by the first two factors and thus it agrees with the findings of part (a). Now, it seems that the first component loads strong for all the variables and can be considered as a measure for the overall subject preferences for college students, where as, for the second component, it loads strong for Algebra, Calculus and Statistics, but behaves as a contrast for the other subjects. So, any suitable overall interpretation can not be found for the second component.

Part (c)

```
pa_mod_norot = fa(data_4,nfactors=2,rotate="none",fm="pa",SMC = T)
pa_mod_norot$PVAL
```

```
## [1] 0.506358
```

```
pa_mod_promax = fa(data_4,nfactors=2,rotate="promax",fm="pa",SMC = T)
```

```
## Loading required namespace: GPArotation
```

```
pa_mod_promax$PVAL
```

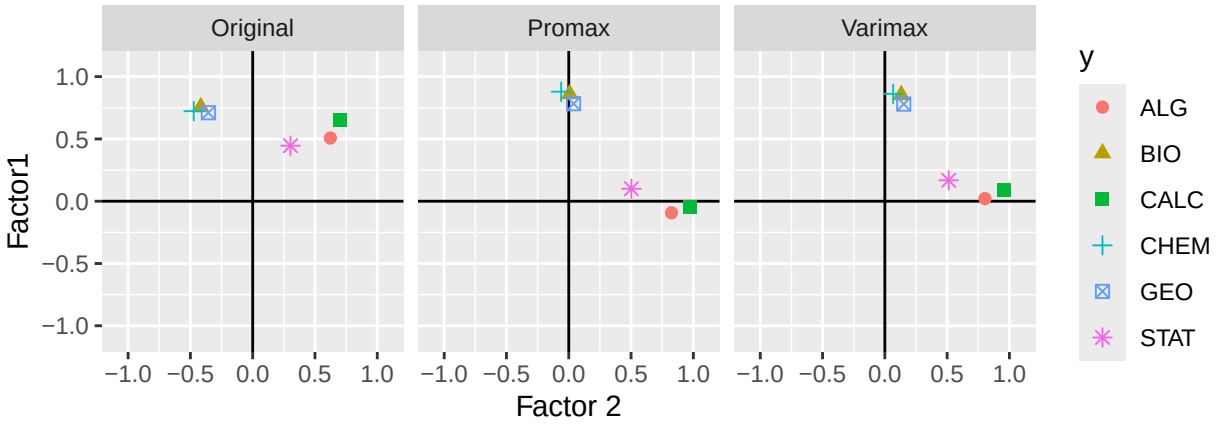
```
## [1] 0.506358
```

```
pa_mod_varimax = fa(data_4,nfactors=2,rotate="varimax",fm="pa",SMC = T)
pa_mod_varimax$PVAL
```

```
## [1] 0.506358
```

```
factors_df = bind_rows(data.frame(y = rownames(pa_mod_norot$loadings),
                                  unclass(pa_mod_norot$loadings)),
                       data.frame(y = rownames(pa_mod_promax$loadings),
                                  unclass(pa_mod_promax$loadings)),
                       data.frame(y = rownames(pa_mod_varimax$loadings),
                                  unclass(pa_mod_varimax$loadings)),
                       .id = "Rotation")

ggplot(factors_df)+
  geom_vline(aes(xintercept=0))+
  geom_hline(aes(yintercept=0))+
  geom_point(aes(x=PA2,y=PA1,col=y,shape=y),size=2)+
  scale_x_continuous(name="Factor 2",limits = c(-1.1,1.1))+
  scale_y_continuous(name="Factor1",limits = c(-1.1,1.1))+
  facet_wrap("Rotation",
             labeller=labeller(Rotation = c("1"="Original","2"="Promax","3"="Varimax")))+
  coord_fixed(ratio=1)
```



From the plots, it is evident that the promax rotation did the best job to make the variables rely on one single component compared to the varimax rotation and no rotation. Between these two, the varimax is considerably better than the one with no rotation.

Part (d)

The analysis indicates that a single factor isn't enough to explain the variability in subject preference. However, with two factors, prove sufficient for the analysis. Attempting to include a third factor isn't feasible due to insufficient data leading to a p-value of NA.

PROBLEM 5.

```
# Loading the train and test data
data_5_train = read.table("C:/Users/Jayaditya Nath/Documents/spamdetect_train.dat",sep = ",")
data_5_train = data_5_train[, -41]

data_5_test = read.table("C:/Users/Jayaditya Nath/Documents/spamdetect_test.dat",sep = ",")
data_5_test = data_5_test[, -41]
```

Part (a)

```
# Extract spam and non-spam groups
spam_group_train = data_5_train[data_5_train[, 57] == 1, -57]
non_spam_group_train = data_5_train[data_5_train[, 57] == 0, -57]

#Testing for equal variance-covariance matrix
CovTest2.2013Cai(as.matrix(spam_group_train),as.matrix(non_spam_group_train),alpha = 0.05)$reject
```

```
## [1] TRUE
```

```
# Test for multivariate normality
mvn(spam_group_train, mvnTest = "hz")$multivariateNormality
```

```
##            Test     HZ p value MVN
## 1 Henze-Zirkler 4.1096       0  NO
```

```
mvn(non_spam_group_train, mvnTest = "hz")$multivariateNormality
```

```
##            Test       HZ p value MVN
## 1 Henze-Zirkler 17.46691       0  NO
```

From the results, it is clear that the covariance matrices for the spam and not spam groups are different and thus, a quadratic discriminant analysis is suggested. Also, the two sets of variables can not be assumed to come from a multivariate normal distribution using the Henze-Zirkler test of multivariate normality.

Part (b)

```
# Fit linear discriminant analysis
lda_mod_5 = lda(as.factor(V58) ~ ., data = data_5_train,prior=c(0.5,0.5))

# Cross-validation
cv_lda = lda(as.factor(V58) ~ ., data = data_5_train, CV = TRUE,prior=c(0.5,0.5))
conf_matrix_cv_lda = table(data_5_train$V58, cv_lda$class)
error_rate_cv_lda = 1 - sum(diag(conf_matrix_cv_lda)) / sum(conf_matrix_cv_lda)

print("The cross-validation confusion matrix is : ")
```

```
## [1] "The cross-validation confusion matrix is : "
```

```
conf_matrix_cv_lda
```

```
##
##        0    1
##   0 2348  136
##   1  244 1373
```

```
print(paste0("The error rate with the train data is : ",error_rate_cv_lda))
```

```
## [1] "The error rate with the train data is : 0.0926603267495733"
```

```
# Predict on test data
pred_lda = predict(lda_mod_5, newdata = data_5_test)

# Confusion matrix and error rate
conf_matrix_lda = table(pred_lda$class, data_5_test$V58)
error_rate_lda = 1 - sum(diag(conf_matrix_lda)) / sum(conf_matrix_lda)

print("The classification confusion matrix is : ")
```

```
## [1] "The classification confusion matrix is : "
```

```
conf_matrix_lda
```

```
##
##       0   1
##   0 282  37
##   1  22 159
```

```
print(paste0("The error rate with the test data is : ",error_rate_lda))
```

```
## [1] "The error rate with the test data is : 0.118"
```

```
# Fit linear discriminant analysis
lda_mod_5 = lda(as.factor(V58) ~ ., data = data_5_train,prior=c(0.5,0.5))

# Cross-validation
cv_lda = lda(as.factor(V58) ~ ., data = data_5_train, CV = TRUE,prior=c(0.5,0.5))
conf_matrix_cv_lda = table(data_5_train$V58, cv_lda$class)
error_rate_cv_lda = 1 - sum(diag(conf_matrix_cv_lda)) / sum(conf_matrix_cv_lda)

print("The cross-validation confusion matrix is : ")
```

```
## [1] "The cross-validation confusion matrix is : "
```

```
conf_matrix_cv_lda
```

```
##
##        0    1
##   0 2348  136
##   1  240 1377
```

```
print(paste0("The error rate with the train data is : ",error_rate_cv_lda))
```

```
## [1] "The error rate with the train data is : 0.0916849548890515"
```

```
# Predict on test data
pred_lda = predict(lda_mod_5, newdata = data_5_test)

# Confusion matrix and error rate
conf_matrix_lda = table(pred_lda$class, data_5_test$V58)
error_rate_lda = 1 - sum(diag(conf_matrix_lda)) / sum(conf_matrix_lda)

print("The classification confusion matrix is : ")
```

```
## [1] "The classification confusion matrix is : "
```

```
conf_matrix_lda
```

```
##
##       0   1
##   0 282  37
##   1  22 159
```

```
print(paste0("The error rate with the test data is : ",error_rate_lda))
```

```
## [1] "The error rate with the test data is : 0.118"
```

The results from using equal prior probabilities(error rate of 12%) are analogous to the results from the discriminant analysis using prior probabilities proportional to sample sizes(error rate of 14%).

Part (c)

```
# Fit quadratic discriminant analysis
qda_mod_5 = qda(as.factor(V58) ~ ., data = data_5_train,prior=c(0.5,0.5))

# Cross-validation
cv_qda = lda(as.factor(V58) ~ ., data = data_5_train, CV = TRUE,prior=c(0.5,0.5))
conf_matrix_cv_qda = table(data_5_train$V58, cv_qda$class)
error_rate_cv_qda = 1 - sum(diag(conf_matrix_cv_qda)) / sum(conf_matrix_cv_qda)

print("The cross-validation confusion matrix is : ")
```

```
## [1] "The cross-validation confusion matrix is : "
```

```
conf_matrix_cv_qda
```

```
##
##        0    1
##   0 2347  137
##   1  241 1376
```

```
print(paste0("The error rate with the train data is : ",error_rate_cv_qda))
```

```
## [1] "The error rate with the train data is : 0.0921726408193123"
```

```
# Predict on test data
pred_qda = predict(qda_mod_5, newdata = data_5_test)

# Confusion matrix and error rate
conf_matrix_qda = table(pred_qda$class, data_5_test$V58)
error_rate_qda = 1 - sum(diag(conf_matrix_qda)) / sum(conf_matrix_qda)


print("The classification confusion matrix is : ")
```

```
## [1] "The classification confusion matrix is : "
```

```
conf_matrix_qda
```

```
##
##       0   1
##   0 208  13
##   1  96 183
```

```
print(paste0("The error rate with the test data is : ",error_rate_qda))
```

```
## [1] "The error rate with the test data is : 0.218"
```

Since the two groups have different covariance matrices, a quadratic discriminant analysis should have performed better, but, here, we can see that the QDA performs worse compared to LDA with an error rate of roughly 22%. This is probably because the multivariate normality assumption is not valid for this data.

Part (d)

```
# Stepwise classification
step_mod_5 = stepclass(V58 ~ ., data = data_5_train, method = "qda", improvement = 0.05)
```

```
##  'stepwise classification', using 10-fold cross-validated correctness rate of method qda'.
```

```
## 4101 observations of 56 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## Warning in cv.rate(vars = c(model, tryvar), data = data, grouping = grouping, :
## error(s) in modeling/prediction step
```

```
## correctness rate: 0.73616;  in: "V53";  variables (1): V53
```

```
## Warning in cv.rate(vars = c(model, tryvar), data = data, grouping = grouping, :
## error(s) in modeling/prediction step
```

```
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00       10.85
```

```
# Fit logistic regression model using the selected predictors
logistic_model = glm(step_mod_5$formula, data = data_5_train, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Make predictions on test data
pred_logistic = predict(logistic_model, newdata = data_5_test, type = "response")

# Confusion matrix and error rate
conf_matrix_qda_step = table(round(pred_logistic), data_5_test$V58)
error_rate_qda_step = 1 - sum(diag(conf_matrix_qda_step)) / sum(conf_matrix_qda_step)

print("The classification confusion matrix is : ")
```

```
## [1] "The classification confusion matrix is : "
```

```
conf_matrix_qda_step
```

```
##
##       0   1
##   0 293  92
##   1  11 104
```

```
print(paste0("The error rate with the test data is : ",error_rate_qda_step))
```

```
## [1] "The error rate with the test data is : 0.206"
```

Only the 53rd variable got selected in the step-wise discriminant analysis and I have chosen an improvement of 5% on the model accuracy since the last model which I fit had an accuracy of roughly 79% and I would not be worried with a change of 5% in the model accuracy.

Till now, the LDA model with equal prior probabilities exhibit the lowest error rate among the QDA and step-wise discriminant analysis models.

Part (e)

```r
# KNN
knn_mod_5 = knn(data_5_train[,-57],data_5_train[,-57],data_5_train[,57],k=3)

knn_mod_5_test = knn(data_5_train[,-57],data_5_test[,-57],data_5_train[,57],k=3)
conf_matrix_knn = table(data_5_test[,57],predict=knn_mod_5_test)


print("The classification confusion matrix is : ")
```

```
## [1] "The classification confusion matrix is : "
```

```r
conf_matrix_knn
```

```
##    predict
##       0   1
##   0 256  48
##   1  55 141
```

```r
print(paste0("The accuracy of the KNN model with the test data is : ",sum(diag(conf_matrix_knn)) / sum(c
```

```
## [1] "The accuracy of the KNN model with the test data is : 0.794"
```

Performing discriminant analysis with the k-nearest neighbours algorithm, I get an error rate of roughly 8%. This model is the best in terms of classification compared to all the models which i have fit.