Multivariate Statistical Analysis Problem set 2

Lorenzo Sala 943481 - Stefano Sperti 947676 - Francesco Virgili 1101739

Università di Torino

```
knitr::opts_chunk$set(echo = TRUE,fig.align="center")
getwd()
set.seed(222)
library(MASS)
library(corrplot)
library(psych)
library(gridExtra)
library(ggplot2)
library(ellipse)
library(randomForest)
```

Exercise 1

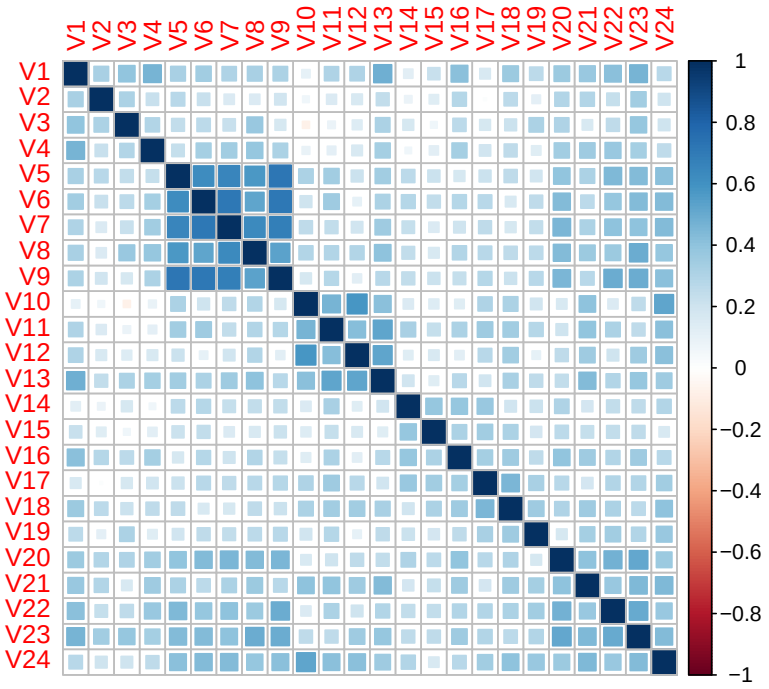
1.1

We use the Grant-White student data. We remove the "Case','Sex" and "Age variable", since our analysis revolves only around psychological tests.

```
grant <- psych[which(psych$group=="GRANT"),4:27]
n<-dim(grant)[1]
p<-dim(grant)[2]
var.names<-names(grant)
```

We scale the dataset so that we will work with standardized values. We also create a correlation plot to better visualize how the (many) variables are correlated with each other.

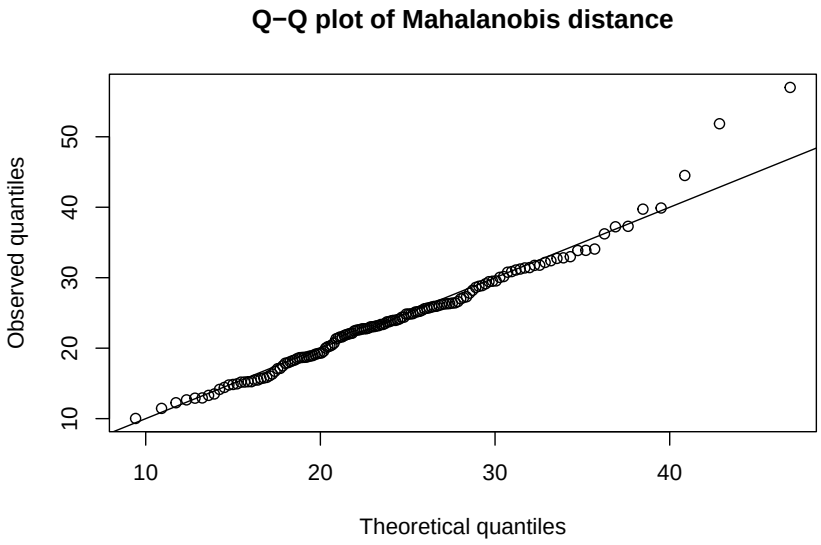
```
# V1 visual perception, V2 cubes, V3 paper form board,
# V4 flags, V5 general information, V6 paragraph comprehension,
# V7 sentence completion, V8 word classification, V9 word meaning,
# V10 addition, V11 code, V12 counting dots,
# V13 straight-curved capitals, V14 word recognition, V15 number recognition,
# V16 figure recognition, V17 object-number, V18 number-figure,
# V19 figure-word, V20 deduction, V21 numerical puzzles,
# V22 problem reasoning, V23 series completion, V24 arithmetic problems.

X <- scale(grant)

# Correlation matrix of the data
R <- cor(X)
corrplot(R,method="square")
```



From the correlation plot we observe that all variable are positively correlated, moreover the plot already highlights that some group of variables are highly correlated between them and less correlated with the other variables. In particular, we can see how the variables $V5$, $V6$, $V7$, $V8$, $V9$ show the highest between correlation. Other -less evident- examples may be found in the groups composed of the variables $V10$, $V11$, $V12$, $V13$ and the one composed by $V20$, $V21$, $V22$, $V23$, $V24$. Another noteworthy observation is the correlation between V24 and V10, which exceeds 0.5. However, considering the need to establish distinct groups and the comparatively lower correlations between V24 and both V13 and V12, it is better to form two separate groups.

We perform tests for normality to check that the assumption on which ML-FA is based holds.

```
dM <- mahalanobis(X,colMeans(X),cov(X))
qqplot(qchisq(ppoints(dM),df=p),dM,xlab = "Theoretical quantiles",
  ylab = "Observed quantiles")
title(
  main = "Q-Q plot of Mahalanobis distance"
  )
abline(0,1)
```



From the above plot, we see that the Mahalanobis distance seems to be χ^2 distributed, apart from the fact that the right tail of the empirical distribution seems to be a little heavier than expected. However, we can still infer that our data are distributed as a multivariate normal distribution. For the rest of our analysis we consider this distibution as Gaussian distibution.

We then display the results of the factor analysis, for now without using any rotation since we will introduce it later on. In particular rotation will affect the distribution of the proportion of the total sample variance explained by each factor.

```
ml5 <- factanal(covmat=R, factors=5,rotation="none")
ml6 <- factanal(covmat=R, factors=6,rotation="none")
cat("Result with 5 factors (ml5):\n")
```

```
## Result with 5 factors (ml5):
```

```
ml5
```

```
##
## Call:
## factanal(factors = 5, covmat = R, rotation = "none")
##
## Uniquenesses:
##    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10   V11   V12   V13
## 0.453 0.777 0.646 0.648 0.357 0.291 0.286 0.481 0.257 0.208 0.413 0.436 0.253
##   V14   V15   V16   V17   V18   V19   V20   V21   V22   V23   V24
## 0.649 0.712 0.565 0.572 0.596 0.761 0.509 0.569 0.568 0.447 0.480
##
## Loadings:
##     Factor1 Factor2 Factor3 Factor4 Factor5
## V1    0.555           0.466  -0.150
## V2    0.344           0.292           0.125
## V3    0.373  -0.142   0.427  -0.104
## V4    0.463  -0.104   0.303  -0.113   0.148
## V5    0.723  -0.254  -0.225
## V6    0.721  -0.374  -0.168          -0.145
## V7    0.728  -0.335  -0.232  -0.132
## V8    0.692  -0.144          -0.107
```

```
## V9   0.723  -0.424  -0.197
## V10  0.518   0.603  -0.379          0.116
## V11  0.570   0.349                 -0.367
## V12  0.487   0.544         -0.118   0.128
## V13  0.631   0.347   0.201 -0.383  -0.206
## V14  0.393                  0.369  -0.238
## V15  0.346           0.128  0.368  -0.128
## V16  0.456           0.378  0.276
## V17  0.453   0.128          0.438  -0.113
## V18  0.475   0.252   0.218  0.259
## V19  0.418           0.138  0.196
## V20  0.596  -0.167   0.181  0.155   0.227
## V21  0.574   0.227   0.154          0.159
## V22  0.595  -0.139   0.180  0.129
## V23  0.665           0.213          0.244
## V24  0.657   0.186  -0.126  0.145   0.129
##
##                Factor1 Factor2 Factor3 Factor4 Factor5
## SS loadings      7.581   1.674   1.316   0.959   0.535
## Proportion Var   0.316   0.070   0.055   0.040   0.022
## Cumulative Var   0.316   0.386   0.440   0.480   0.503
##
## The degrees of freedom for the model is 166 and the fit was 1.3988
```

```
cat("Result with 6 factors (ml5):\n")
```

```
## Result with 6 factors (ml5):
```

```
ml6
```

```
##
## Call:
## factanal(factors = 6, covmat = R, rotation = "none")
##
## Uniquenesses:
##    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10   V11   V12   V13
## 0.447 0.710 0.577 0.643 0.346 0.295 0.252 0.429 0.248 0.216 0.311 0.426 0.289
##   V14   V15   V16   V17   V18   V19   V20   V21   V22   V23   V24
## 0.693 0.732 0.586 0.347 0.586 0.758 0.513 0.523 0.549 0.449 0.485
##
## Loadings:
##     Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
## V1   0.549           0.456  -0.197
## V2   0.339           0.301  -0.158           0.232
## V3   0.373  -0.139   0.444  -0.111          -0.232
## V4   0.460  -0.107   0.304  -0.133   0.126
## V5   0.724  -0.261  -0.217
## V6   0.724  -0.367  -0.156          -0.146
## V7   0.733  -0.354  -0.234                  -0.148
## V8   0.695  -0.155         -0.105          -0.207
## V9   0.728  -0.421  -0.180
## V10  0.513   0.587  -0.385           0.160
## V11  0.579   0.390                  -0.422   0.127
## V12  0.482   0.536          -0.165   0.128  -0.103
## V13  0.618   0.328   0.153  -0.357  -0.228  -0.140
## V14  0.398                   0.353  -0.131
```

```
## V15  0.349           0.146   0.332
## V16  0.457           0.388   0.210
## V17  0.474   0.180           0.570          -0.257
## V18  0.478   0.278   0.233   0.221
## V19  0.422           0.154   0.184
## V20  0.596  -0.156   0.201           0.231
## V21  0.571   0.232   0.151           0.137   0.216
## V22  0.597  -0.121   0.198                   0.169
## V23  0.662           0.229           0.226
## V24  0.656   0.190  -0.113           0.158
##
##                Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
## SS loadings      7.602   1.707   1.351   1.000   0.509   0.419
## Proportion Var   0.317   0.071   0.056   0.042   0.021   0.017
## Cumulative Var   0.317   0.388   0.444   0.486   0.507   0.525
##
## The degrees of freedom for the model is 147 and the fit was 1.1998
```

```
L5 <- ml5$loadings
L6 <- ml6$loadings
```

Since the proportion of total sample variance caused by a certain factor is defined as

$$\frac{\sum_{j=1}^{p} \hat{\ell}_{jk}}{\text{trace}(\mathbf{S})}$$

where $\hat{\ell}_{jk}$ is an entry of the $\hat{\mathbf{L}}$ matrix containing the estimation of the loadings, we can obtain the contribution to each factor by the sum of the squared loadings (whose result coincides with the corresponding output of the factor analysis command).

```
cat("The variance explained by each factor when m=5 is:\n")
```

```
## The variance explained by each factor when m=5 is:
```

```
round(colSums(L5^2)/tr(R), 3)
```

```
## Factor1 Factor2 Factor3 Factor4 Factor5
##   0.316   0.070   0.055   0.040   0.022
```

```
cat("The variance explained by each factor when m=6 is:\n")
```

```
## The variance explained by each factor when m=6 is:
```

```
round(colSums(L6^2)/tr(R), 3)
```

```
## Factor1 Factor2 Factor3 Factor4 Factor5 Factor6
##   0.317   0.071   0.056   0.042   0.021   0.017
```

In both cases, we see that the first three factors are the ones that explain the greatest proportion of variance, with factors 4, 5 (and 6) showing little contribution in explaining the overall variance. In both cases, it slightly exceeds 50%, resulting in a total increase from 50.3% to 52.5%. In particular, we can observe that the proportion of variance explained by the first factor is greater with respect to all the other ones.

List the specific variances, and assess the accuracy of the approximation of the correlation matrix. Compare the results. Which choice of m do you prefer? Why?

We calculate the MLE estimate of communalities as

$$\hat{h}_j^2 = \hat{\ell}_{j1}^2 + \ldots + \hat{\ell}_{jm}^2 \qquad \text{for } j = 1, \ldots, p$$

```r
# Communalities
h5 <- rowSums(L5^2)
h6 <- rowSums(L6^2)
```

We then calculate the matrix of specific variances $\psi_j = 1 - h_j$. Another possible solution is to use the command $uniquenesses that display the specific variances.

```r
# Specific variances
psi5 <- 1-h5
cat("Specific variance for m=5:\n")
```

```
## Specific variance for m=5:
```

```r
ml5$uniquenesses
```

```
##        V1        V2        V3        V4        V5        V6        V7        V8
## 0.4526292 0.7766316 0.6456232 0.6477263 0.3572546 0.2907279 0.2862756 0.4811690
##        V9       V10       V11       V12       V13       V14       V15       V16
## 0.2573406 0.2082378 0.4133526 0.4360737 0.2525149 0.6488806 0.7117565 0.5653934
##       V17       V18       V19       V20       V21       V22       V23       V24
## 0.5723682 0.5960215 0.7607123 0.5086459 0.5694681 0.5682620 0.4474393 0.4798638
```

```r
psi6 <- 1-h6
cat("Specific variance for m=6:\n")
```

```
## Specific variance for m=6:
```

```r
ml6$uniquenesses
```

```
##        V1        V2        V3        V4        V5        V6        V7        V8
## 0.4474225 0.7098033 0.5771202 0.6433497 0.3459739 0.2946313 0.2519095 0.4288371
##        V9       V10       V11       V12       V13       V14       V15       V16
## 0.2481073 0.2164355 0.3110665 0.4261631 0.2885378 0.6925193 0.7319866 0.5863738
##       V17       V18       V19       V20       V21       V22       V23       V24
## 0.3472614 0.5857386 0.7583107 0.5127934 0.5232927 0.5491322 0.4494480 0.4852731
```

The specific variances can help evaluate the overall fit of the factor model. Large specific variances may indicate poor model fit and the need for modifications. Conversely, small specific variances may suggest overfitting and the need for simplification. In our case, the presence of many variables with high uniqueness suggests a poor fit of the model to the data. Additionally, only two variables show significantly reduced uniqueness with the introduction of a sixth factor, implying that the six-factor model may not offer substantial improvement over the five-factor model.

To choose the number of factors m we can compare the residual matrices

$$\mathbf{R} - \left(\hat{\mathbf{L}}\hat{\mathbf{L}}^T + \hat{\mathbf{\Psi}}\right).$$

where $\hat{\mathbf{L}}$ is the matrix of estimated loadings and $\hat{\mathbf{\Psi}}$ is the diagonal matrix containing $\hat{\psi}_j = s_{jj} - \sum_{k=1}^{m} \hat{\ell}_{jk}^2$

```r
# Residual matrix
Res5 <- R-L5%*%t(L5)-diag(ml5$unique)
cat("Frobenius norm residual matrix m=5: \n")
```

```
## Frobenius norm residual matrix m=5:
```

```r
sum(Res5^2)
```

```
## [1] 0.7335059
```

```
Res6 <- R-L6%*%t(L6)-diag(ml6$unique)
cat("Frobenius norm residual matrix m=6: \n")
```

```
## Frobenius norm residual matrix m=6:
```

```
sum(Res6^2)
```

```
## [1] 0.6020222
```

As we can see, residuals obtained with 6 factors are less than residuals obtained with 5 factors: this meas that the maximum likelihood estimates $\hat{\mathbf{L}}_6$ and $\hat{\mathbf{\Psi}}_6$ reproduce the original $\mathbf{R}$ matrix better than their counterparts $\hat{\mathbf{L}}_5$ and $\hat{\mathbf{\Psi}}_5$. Choosing $m = 6$, however, comes at a cost:

```
(6+1)*p-(5+1)*p
```

```
## [1] 24
```

we would have to estimate 24 new parameters just to obtain a negligible improvement in the proportion of variance explained by the model (+1,7%), thus using a sort of "parsimony" principle $m = 5$ seems like a better choice.

1.2

Give an interpretation to the common factors in the $m = 5$ solution with varimax rotation.

We know that loadings are defined up to a rotation of the rows of the matrix $\mathbf{L}$. To ease interpretation, it is often useful to apply a rotation to factor so that each data point loads the most on few factors. We employ the *varimax* rotation, which seek the rotated loadings $\hat{\mathbf{L}}^* = \hat{\mathbf{L}}\mathbf{T}$ that maximize the variance of the squared loadings for each factor, for $\mathbf{T}$ that maximizes

$$V = \sum_{k=1}^{m} \left[\frac{1}{p} \sum_{j=1}^{p} (\tilde{\ell}_{jk}^*)^4 - \left(\frac{1}{p} \sum_{j=1}^{p} (\tilde{\ell}_{jk}^*)^2 \right)^2 \right]$$

where $\tilde{\ell}_{jk}^* = \frac{\hat{\ell}_{jk}^*}{\hat{h}_j^{1/2}}$ are the rotated and scaled loadings of the k-th factor.

```
ml5varimax <- factanal(covmat=R, factors=5,rotation="varimax")
L5=ml5varimax$loadings
L5
```

```
##
## Loadings:
##     Factor1 Factor2 Factor3 Factor4 Factor5
## V1   0.165   0.655   0.125   0.181   0.207
## V2   0.108   0.442
## V3   0.134   0.559           0.112
## V4   0.230   0.533
## V5   0.738   0.189   0.192   0.149
## V6   0.772   0.187           0.248   0.124
## V7   0.798   0.214   0.143
## V8   0.571   0.343   0.239   0.128
## V9   0.808   0.202           0.219
## V10  0.181  -0.108   0.845   0.180
## V11  0.195           0.423   0.436   0.418
## V12          0.232   0.694   0.102   0.129
## V13  0.186   0.433   0.479           0.538
## V14  0.185                   0.552
```

```
## V15  0.104   0.122          0.509
## V16          0.406          0.509
## V17  0.154          0.210   0.595
## V18          0.300   0.322   0.458
## V19  0.156   0.221   0.144   0.378
## V20  0.373   0.461   0.127   0.293  -0.194
## V21  0.172   0.398   0.431   0.238
## V22  0.364   0.423   0.114   0.320
## V23  0.362   0.542   0.248   0.231  -0.115
## V24  0.368   0.179   0.495   0.321
##
##                Factor1 Factor2 Factor3 Factor4 Factor5
## SS loadings      3.640   2.957   2.454   2.386   0.628
## Proportion Var   0.152   0.123   0.102   0.099   0.026
## Cumulative Var   0.152   0.275   0.377   0.477   0.503
```

We establish a threshold of 0.45 to detect the most relevant (i.e. which have the heaviest loadings) variables for each factor. We sort the factor loading in decreasing order so that our output will present the variables in order of "importance". We start with the first factor:

```
names(which(sort(L5[,1],T)>0.45))
```

```
## [1] "V9" "V7" "V6" "V5" "V8"
```

Here:

- $V9$ is the "word meaning" variable.
- $V7$ is the "sentence completion" variable;
- $V6$ is the "paragraph comprehension" variable;
- $V5$ is the "general information" variable;
- $V8$ is the "word classification" variable;

It seems intuitive to name "linguistic abilities" a factor which is heavily loading on such variables. It is worth noting that these variables correspond to the most evident group that was pointed out from the correlation plot in section 1.1.

For the second factor we have:

```
names(which(sort(L5[,2],T)>0.45))
```

```
## [1] "V1"  "V3"  "V23" "V4"  "V20"
```

Here:

- $V1$ is the "visual perception" variable;
- $V3$ is the "paper form board" variable;
- $V23$ is the "series completion" variable;
- $V4$ is the "flags" variable;
- $V20$ is the "deduction" variable.

Here the factor is more tricky to interpret: we could call this "spatial-logical abstraction", given that the tests involve a mix of pattern recognition and abstraction skills. It is also worth noting that the two variables that concur the most to the factor are related to the visual range.

We examine the third factor:

```r
names(which(sort(L5[,3],T)>0.45))
```

```
## [1] "V10" "V12" "V24" "V13"
```

Here:

- $V10$ is the "addition" variable;
- $V12$ is the "counting dots" variable;
- $V24$ is the "arithmetic problems" variable;
- $V13$ is the "straight curved capitals" variable.

Beside $V13$ (that is the least signficative variable among the group anyway), we could summarize this factor as "numerical abilities" of the students.

The following two factors are more difficult when it comes to assigning a meaning, however their interpretation is not that significative since the amount of variance that they explain is relatively small.

```r
names(which(sort(L5[,4],T)>0.45))
```

```
## [1] "V17" "V14" "V15" "V16" "V18"
```

In the fourth factor:

- $V17$ is the "object-number" variable;
- $V14$ is the "word recognition" variable;
- $V15$ is the "number recognition" variable;
- $V16$ is the "figure recognition" variable;
- $v18$ is the "number-figure" variable.

We could interpret this factor as the "pattern recognition" factor.

We note that the fifth factor explains a proportion of variance significantly smaller than the other four. Indeed, fifth factor has generally low loadings, so we lower our threshold to include more variables and hopefully gain more insight on what it means.

```r
names(which(sort(L5[,5],T)>0.4))# V11 code, V13 straight-curved capitals
```

```
## [1] "V13" "V11"
```

Here:

- $V13$ is the "straight-curved capitals" variable;
- $V11$ is the "code" variable.

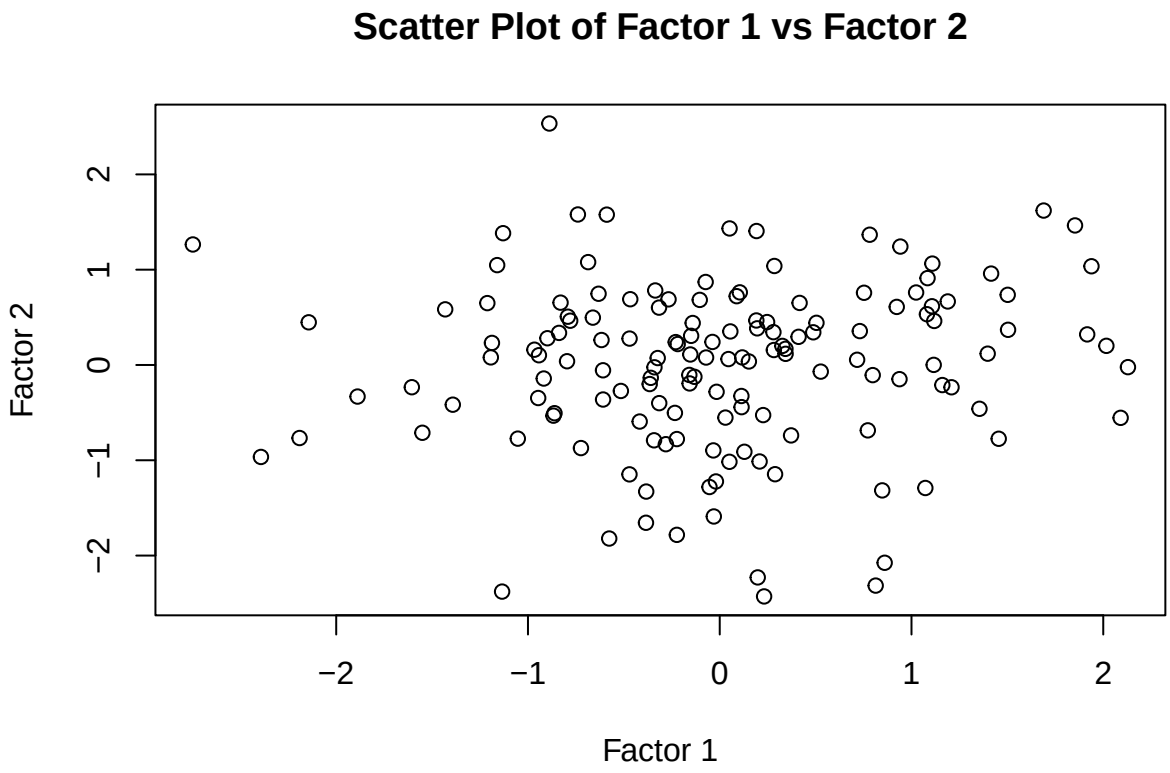
It is hard to assign a meaning to this factor. Maybe we could interpret it as some kind of logical reasoning relative to the field of written production. This might be attributed to the observed variance, which is merely 0.026. Consequently, this could indicate that the fifth factor lacks a clear significance as it appears to be inconsequential.

1.3

Make a scatterplot of the first two factor scores for the $m = 5$ solution obtained by the regression method. Is their correlation equal to zero? Should we expect so? Comment. We start plotting a scatterplot

```r
fareg <- factanal(X,factors=5,scores="regression",rotation = "varimax")
plot(fareg$scores[,1], fareg$scores[,2],
```

```
     xlab = "Factor 1", ylab = "Factor 2",
     main = "Scatter Plot of Factor 1 vs Factor 2")
```



No clear pattern is observed, and for deriving a quantitative indicator, we estimate the factor scores using the regression method by specifying scores="regression" with the ML method.

```
cor(fareg$scores[,1],fareg$scores[,2])
```

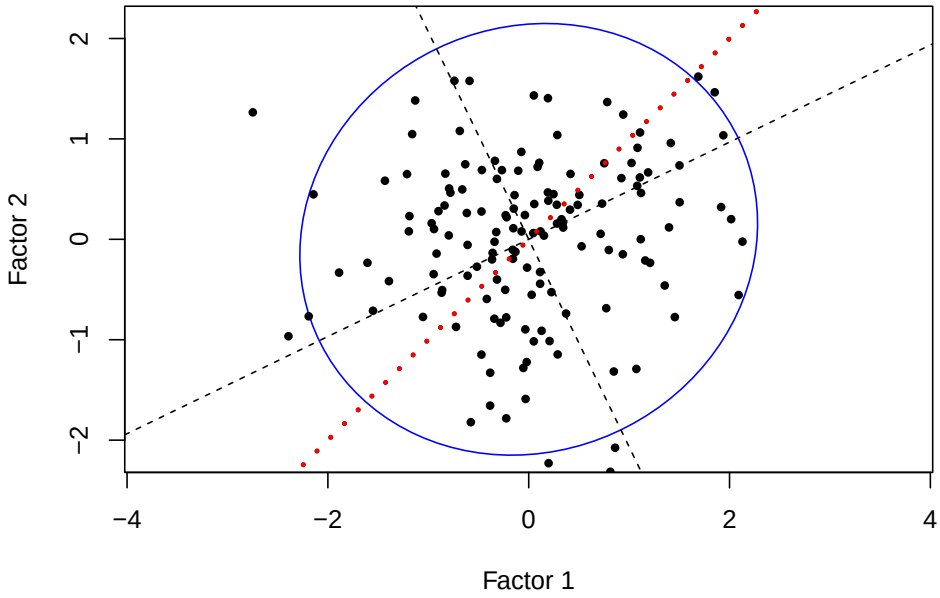
```
## [1] 0.07425218
```

Computing the correlation, we observe that it is almost equal to zero.

We create also a scatterplot of the first two factor scores with the 95% ellipse.

```
F1 <- (fareg$scores)[,1]
F2 <- (fareg$scores)[,2]
r_fareg=cov(fareg$scores[,1:2])
barx_fareg=colMeans(fareg$scores[,1:2])
eigen_fareg=eigen(r_fareg,symmetric = T)
plot(ellipse(x = r_fareg, centre = barx_fareg, level = 0.95),
    col = "blue", type = "l", asp = 1,
    xlab = "Factor 1", ylab = "Factor 2")
points(F1, F2, pch = 16,  cex = 0.75)
b = -eigen_fareg$vectors[1, 2] / eigen_fareg$vectors[2, 2]
a = -b * barx_fareg[1] + barx_fareg[2]
abline(a, b, lwd = 1, lty = 2)
d = -eigen_fareg$vectors[1, 1] / eigen_fareg$vectors[2, 1]
c = -d * barx_fareg[1] + barx_fareg[2]
```

```
abline(c, d, lwd = 1, lty = 2)
abline(a=0,b=1,lty=3,lwd=3)
abline(a=0,b=1,lty=3,lwd=3,col = "red")
```

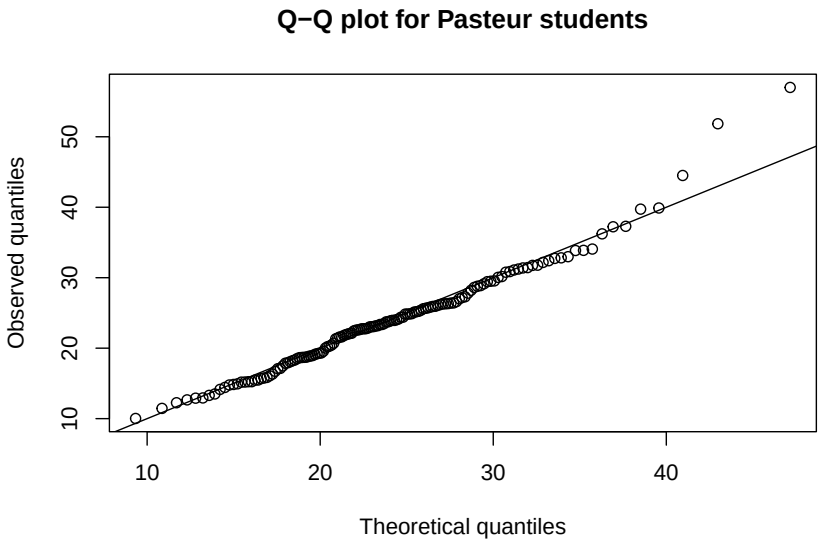


The correlation is low as expected, since the factor analyisis performed with maximum likelihood assumes the non correlation of the factors. Data points assume a wide elliptical shape, as expected, and the fact that the 95% normal ellipsoid is almost a circle further confirms that the first and the second factors are uncorrelated.

1.4

Obtain the maximum likelihood solution with varimax rotation for $m = 5$ factors by using the Pasteur students data. Is the interpretation to the common factors similar to that of Grant–White students? We now compute the maximum likelihood solution with varimax rotation for $m = 5$ factors by using the Pasteur students data. First of all we check if normality holds:

```
pasteur <- psych[which(psych$group=="PASTEUR"),4:27]
Y <- scale(pasteur)
Ry <- cor(Y)
dM2 <- mahalanobis(Y,colMeans(Y),cov(Y))
qqplot(qchisq(ppoints(dM2),df=p),dM,xlab = "Theoretical quantiles",
  ylab = "Observed quantiles")
title(
  main = "Q-Q plot for Pasteur students"
  )
abline(0,1)
```



The qq-plot of the Mahalanobis distance is paractically identical to the previous one, obtained with the Grant-White data, so once again we assume the normality of the data to hold. We perform the data analysis using the varimax rotation

```
ml5y <- factanal(covmat=Ry, factors=5,rotation="varimax")
L5y <- ml5y$loadings
L5y
```

```
##
## Loadings:
##     Factor1 Factor2 Factor3 Factor4 Factor5
## V1   0.314   0.578   0.138
## V2           0.517                  -0.144
## V3           0.444  -0.177
## V4           0.671   0.190   0.170
## V5   0.806                           0.143
## V6   0.782   0.157                   0.202
## V7   0.904                   0.109
## V8   0.684   0.169   0.139   0.152
## V9   0.775   0.249           0.102   0.156
## V10  0.141  -0.208   0.116   0.500   0.641
## V11  0.349           0.231   0.671
## V12                          0.526   0.217
## V13          0.272           0.544
## V14                  0.690
## V15 -0.133   0.125   0.613  -0.110
## V16          0.386   0.475   0.176   0.161
## V17                  0.523   0.289
## V18  0.100           0.465
## V19          0.244   0.357   0.241
## V20  0.123   0.514   0.189
## V21  0.284   0.387   0.141   0.195   0.432
## V22  0.469   0.481           0.152
## V23  0.357   0.587   0.144           0.299
## V24  0.218   0.294   0.226   0.240   0.530
```

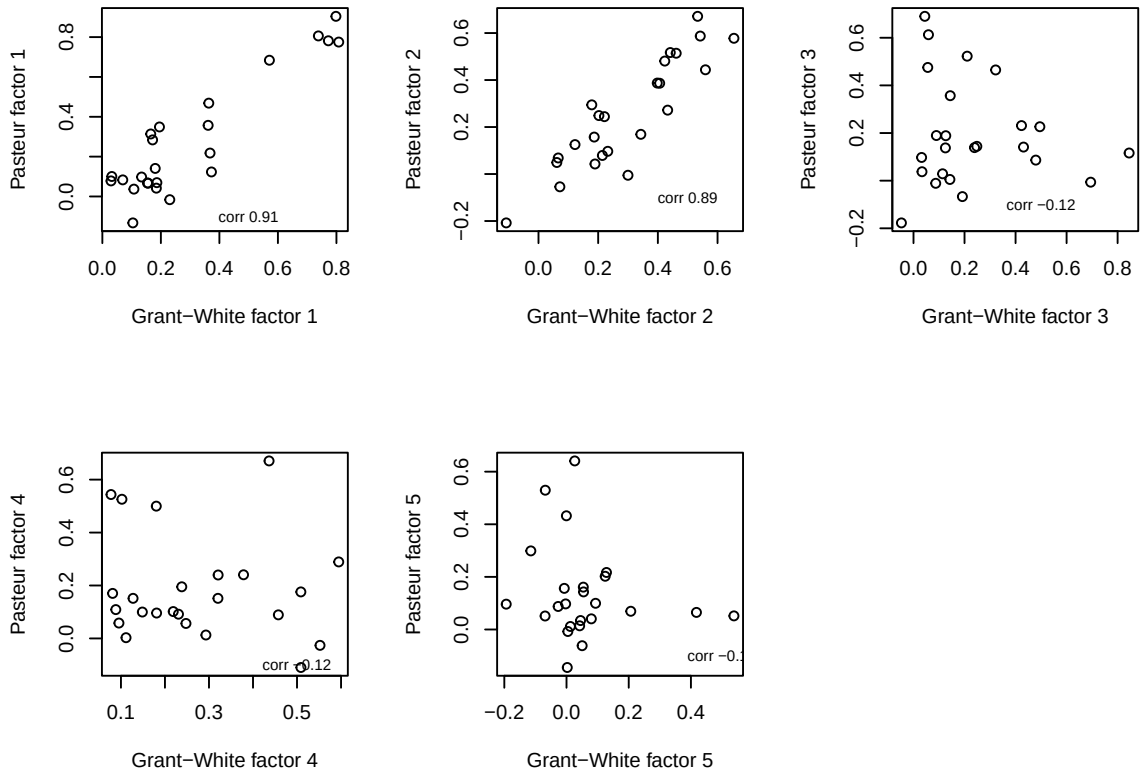
```
##
##                Factor1 Factor2 Factor3 Factor4 Factor5
## SS loadings      3.944   2.810   2.018   1.691   1.205
## Proportion Var   0.164   0.117   0.084   0.070   0.050
## Cumulative Var   0.164   0.281   0.366   0.436   0.486
```

We create a series of plots to examine how these new factors behave with respect to the Grant ones.

```
par(mfrow=c(2,3))
for (i in 1:5) {
  plot(L5[,i],L5y[,i],
       ylab = paste("Pasteur factor",i),
       xlab = paste("Grant-White factor",i))
  lab <- round(cor(L5[,i],L5y[,i]),2)
  text(x=0.5,y=-0.1,labels=paste("corr",lab,sep=" "),cex=0.7)
}

ml5y
```

```
##
## Call:
## factanal(factors = 5, covmat = Ry, rotation = "varimax")
##
## Uniquenesses:
##    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10   V11   V12   V13
## 0.535 0.707 0.752 0.484 0.314 0.311 0.160 0.461 0.301 0.263 0.366 0.661 0.616
##   V14   V15   V16   V17   V18   V19   V20   V21   V22   V23   V24
## 0.517 0.570 0.561 0.628 0.766 0.750 0.675 0.525 0.523 0.409 0.477
##
## Loadings:
##     Factor1 Factor2 Factor3 Factor4 Factor5
## V1   0.314   0.578   0.138
## V2           0.517                  -0.144
## V3           0.444  -0.177
## V4           0.671   0.190   0.170
## V5   0.806                           0.143
## V6   0.782   0.157                   0.202
## V7   0.904                   0.109
## V8   0.684   0.169   0.139   0.152
## V9   0.775   0.249           0.102   0.156
## V10  0.141  -0.208   0.116   0.500   0.641
## V11  0.349           0.231   0.671
## V12                          0.526   0.217
## V13          0.272           0.544
## V14                  0.690
## V15 -0.133   0.125   0.613  -0.110
## V16          0.386   0.475   0.176   0.161
## V17                  0.523   0.289
## V18  0.100           0.465
## V19          0.244   0.357   0.241
## V20  0.123   0.514   0.189
## V21  0.284   0.387   0.141   0.195   0.432
## V22  0.469   0.481           0.152
## V23  0.357   0.587   0.144           0.299
## V24  0.218   0.294   0.226   0.240   0.530
```

```
##
##                Factor1 Factor2 Factor3 Factor4 Factor5
## SS loadings      3.944   2.810   2.018   1.691   1.205
## Proportion Var   0.164   0.117   0.084   0.070   0.050
## Cumulative Var   0.164   0.281   0.366   0.436   0.486
##
## The degrees of freedom for the model is 166 and the fit was 1.1933
```



We see that there is a substantial correlation between the first and second factor for both dataset, while the other three seems pretty different. Indeed the interpretation of the first factor remains practically unchanged.

```
names(which(sort(L5y[,1],T)>0.45))
```

```
## [1] "V7"  "V5"  "V6"  "V9"  "V8"  "V22"
```

- $V7$ is the "sentence completion" variable;
- $V5$ is the "general information" variable;
- $V6$ is the "paragraph comprehension" variable;
- $V9$ is the "word meaning" variable;
- $V8$ is the "word classification" variable;
- $V22$ is the "problem reasoning" variable.

We can see how the first Pasteur factor includes the same variables of the Grant factor analysis. The difference lies in the presence of the variable $V22$ (which is, however, the least important among the six variables

selected) and in the ordering of variables, whose importance is now different. We can conclude that the interpretation of the factor remains the same.

About the second factor:

```
names(which(sort(L5y[,2],T)>0.45))
```

```
## [1] "V4"  "V23" "V1"  "V2"  "V20" "V22"
```

Here:

- $V4$ is the "flags" variable;
- $V23$ is the "series completion" variable;
- $V1$ is the "visual perception" variable;
- $V3$ is the "paper form board" variable;
- $V2$ is the "cube" variable;
- $V20$ is the "deduction" variable.
- $V22$ is the "problem reasoning" variable.

We see a similar situation to the first factor, with a different ordering and more variables included with respect to the Grant data. We can conclude that here, as well, the interpretation can be similar.

About the third one:

```
names(which(sort(L5y[,3],T)>0.45))
```

```
## [1] "V14" "V15" "V17" "V16" "V18"
```

Here:

- $V14$ is the "word recognition" variable;
- $V15$ is the "number recognition" variable;
- $V17$ is the "object-number" variable;
- $V16$ is the "figure recognition" variable;
- $V18$ is the "number-figure" variable.

This factor is radically different from the third factor of the Grant data, instead it loads on exactly the same variables of the fourth factor in the previous dataset. Similarly the fourth factor for the Pasteur dataset loads mainly on:

```
names(which(sort(L5y[,4],T)>0.45))
```

```
## [1] "V11" "V13" "V12" "V10"
```

- $V11$ is the "code" variable;
- $V13$ is the "straight-curved capitals" variable;
- $V12$ is the "counting dots" variable;
- $V10$ is the "addition" variable.

Thus is very similar to the factor that was the third in the previous model.

For the 5th factor the main variables are:

```
names(which(sort(L5y[,5],T)>0.45))
```

```
## [1] "V10" "V24"
```

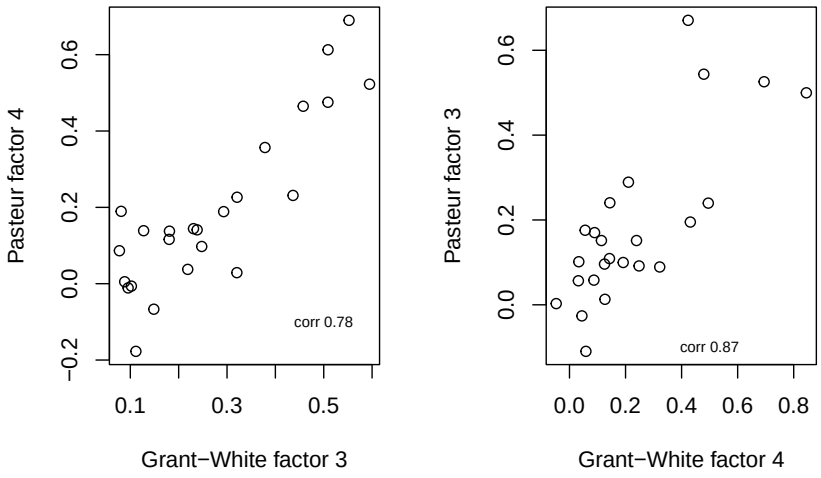
where:

- $V10$ is the "addition" variable;
- $V24$ is the "arithmetic problems" variable.

The fifth factor in the Pasteur model shows no resemblance to the fifth factor in the Grant-White model but this fact has limited relevance, considering the little portion of variance explained by the factor in both models.

In summary, the two models have basically the same factors but the third and fourth one are inverted in the order because one explains more variance than the other in one model and viceversa. Indeed we can see that the correlation pattern which was evident for the first two factors, still appears between factor 3 and 4 and 4 and 3:

```
par(mfrow=c(1,2))
plot(L5[,4],L5y[,3],
       ylab = paste("Pasteur factor",4),
       xlab = paste("Grant-White factor",3))
lab <- round(cor(L5[,3],L5y[,4]),2)
text(x=0.5,y=-0.1,labels=paste("corr",lab,sep=" "),cex=0.7)
plot(L5[,3],L5y[,4],
       ylab = paste("Pasteur factor",3),
       xlab = paste("Grant-White factor",4))
lab <- round(cor(L5[,4],L5y[,3]),2)
text(x=0.5,y=-0.1,labels=paste("corr",lab,sep=" "),cex=0.7)
```



We see that the two fifth factors are moderately correlated.

Overall we can deem the interpretation of the factors as very similar, which means that the solution is stable. Among the differences, we have the slightly different proportion of variance explained by the factors (respectively 0.152 0.123 0.102 0.099 0.026 and 0.164 0.117 0.084 0.070 0.050).

Exercise 2 Let us consider the dataset *pendigits* which was created by collecting 250 samples from 44 writers. These writers were asked to write 250 digits in random order inside boxes of 500 by 500 tablet pixel resolution. The raw data on each of n = 10992 handwritten digits consisted of a sequence, (x_t, y_t), $t = 1, 2, ..., T$, of tablet coordinates of the pen at fixed time intervals of 100 milliseconds, where x_t and y_t were integers in the range 0-500. These data were then normalized to make the representations invariant to translation and scale distortions. The new coordinates were such that the coordinate that had the maximum range varied between 0 and 100. Usually x_t stays in this range, because most integers are taller than they are wide. Finally, from the normalized trajectory of each handwritten digit, 8 regularly spaced measurements, (x_t, y_t), were chosen by spatial resampling, which gave a total of p = 16 variables. The data includes a class attribute, column digit, coded $0, 1, ..., 9$, about the actual digit.

```
#setwd("C:/Users/fravi/Desktop/Universita/Laurea Magistrale/Anno 1/2° semestre/Multivariate statistical
pendigits<-read.table("data/pendigits.txt", sep=",",head=F)
names(pendigits)<-c(paste0(rep(c("x","y"),8),rep(1:8,each=2)),"digit")
n <- dim(pendigits)[1]
p <- dim(pendigits)[2]-1
head(pendigits)
```

```
##    x1  y1 x2  y2  x3  y3  x4  y4 x5 y5  x6 y6  x7 y7  x8 y8 digit
## 1  47 100 27  81  57  37  26   0  0 23  56 53 100 90  40 98     8
## 2   0  89 27 100  42  75  29  45 15 15  37  0  69  2 100  6     2
## 3   0  57 31  68  72  90 100 100 76 75  50 51  28 25  16  0     1
## 4   0 100  7  92   5  68  19  45 86 34 100 45  74 23  67  0     4
## 5   0  67 49  83 100 100  81  80 60 60  40 40  33 20  47  0     1
## 6 100 100 88  99  49  74  17  47  0 16  37  0  73 16  20 20     6
```

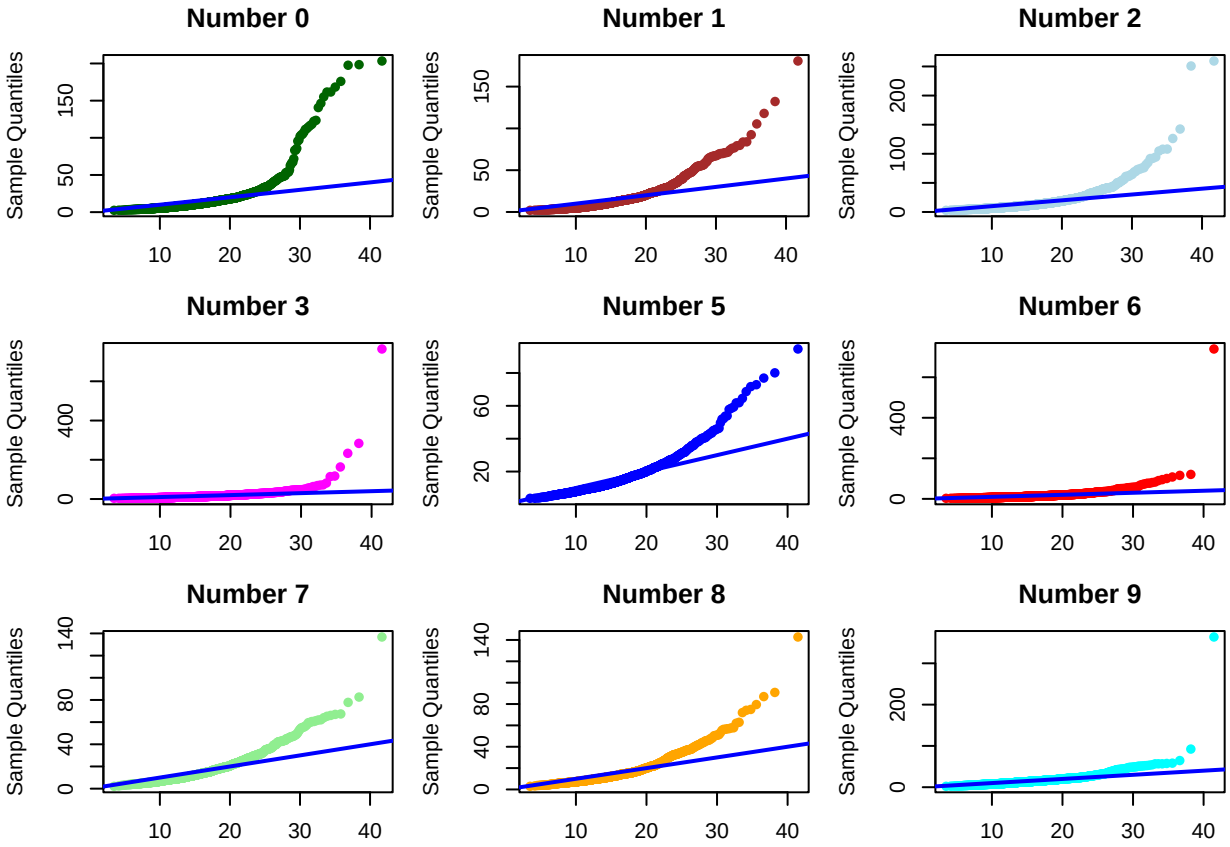
2.1

Use linear discriminant analysis (LDA). Display the first two LD variables in a scatterplot, color coding the observations according to the digit class (use lookup color vector below). How well do they discriminate the 10 digits? Refer also to theory.

```
lookup<-c("darkgreen", "brown", "lightblue", "magenta", "purple",
          "blue", "red", "lightgreen", "orange", "cyan")
```

In Linear Discriminant Analysis (LDA) the goal is to separate observations into classes and assign new observations accordingly. Let us now define, $\mathcal{G} = \{0, 1, .., 9\}$, 10 population classes, each corresponding to digits from 0 to 9. We denote by $f_k(x)$ the class-conditional density and p_k the prior probability for class $\mathcal{G}_k$. The classification of new observations involves maximizing the posterior probability $P(\mathcal{G}_k|X = x)$, conditioned on observing x, which can be expressed using Bayes' rule. LDA arises in the special case when we assume that the classes follow a multivariate Gaussians with a common covariance matrix Σ. So first we check the gaussianity

```
oneDigits <- split(pendigits[, 1:16], pendigits[, 17])
par(mfrow = c(3, 3), mar = c(2, 4, 3, 1))
for (i in 1:10) {
  if (i != 5){
  d <- mahalanobis( oneDigits[[i]], center = colMeans( oneDigits[[i]]),
                    cov = cov( oneDigits[[i]]))
  plot(qchisq(ppoints(d), df = ncol( oneDigits[[i]])), sort(d), pch = 16,
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       main = paste0("Number ", (i - 1)),col=lookup[i])
  abline(0, 1, col = "blue",lw = 2)
}
}
```



The Mahalanobis distance relative to each class does not seem to be χ^2 distributed because each time the qq-plot shows an heavier tail for the empirical distribution, which suggests that the assumption of normality for the classes is not very strong. Notice that we have excluded the class relative to digit 4 since its last variable (y_8) is constantly 0, therefore, it is certainly not Gaussian.

We also check the other assumption of LDA relative to the homoschedasticity of the classes, using Bartlett's test.

```
#names(train)
#c(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8)
for (i in 1:16) {
  res <- bartlett.test(x1~digit,data=pendigits)
  if (res$p.value<0.05) {
    print(paste("The variance of",names(pendigits)[i],"is different between the 10 groups",
                "p-value of Bartlett's test: ",res$p.value,sep=" "))
  }
}
```

```
## [1] "The variance of x1 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y1 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of x2 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y2 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of x3 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y3 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of x4 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y4 is different between the 10 groups p-value of Bartlett's test:  0"
```

```
## [1] "The variance of x5 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y5 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of x6 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y6 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of x7 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y7 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of x8 is different between the 10 groups p-value of Bartlett's test:  0"
## [1] "The variance of y8 is different between the 10 groups p-value of Bartlett's test:  0"
```

For each variable, we observe that the Bartlett test rejects the hypothesis of equal variance between groups.

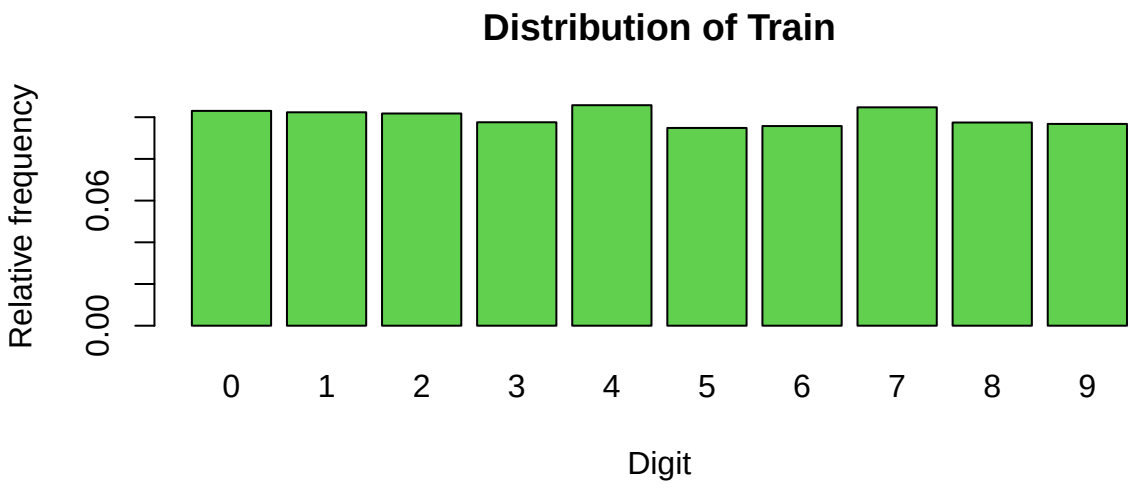
Even though the distributional assumptions are not met, the Linear Discriminant Analysis (LDA) algorithm remains relevant. Although adhering to these assumptions yields optimal performance, LDA still offers valuable insights and efficient classification. LDA iteratively discerns discriminant features from the ten classes, aiming to maximize inter-class separation while minimizing intra-class variance.

To assess our model, we divide the dataset in a training set, to train the model, and a test set, to evaluate its performance. In particular, we've chosen to sample a train set from with size equal to 80% of the size of the original dataset:

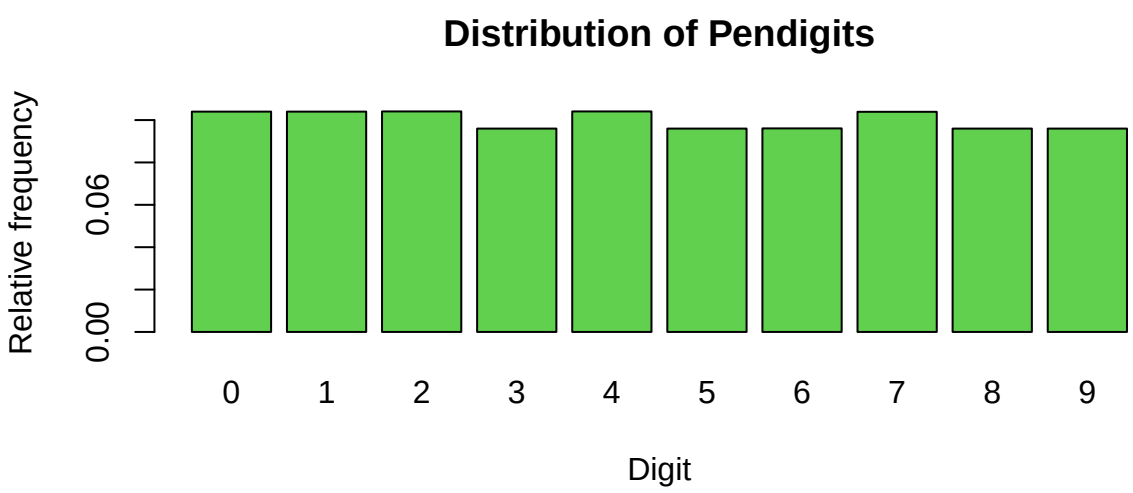
```
nt <- round(n*0.8)
ind <- sample(1:n,nt)
train <- pendigits[ind,]
test <- pendigits[-ind,]
col.index <- lookup[(train$digit)+1]
```

Let us now verify that the distribution of the classes is uniform across the dataset by examining an histogram. This will ensure that all classes are equaly represented in the training set.

```
barplot(table(train$digit)/nt, main = "Distribution of Train",
        xlab = "Digit", ylab = "Relative frequency",col=3)
```




```
barplot(table(pendigits$digit)/n, main = "Distribution of Pendigits",
        xlab = "Digit", ylab = "Relative frequency",col=3)
```



We can observe that the distribution is almost the same, so our train set well represents the original data. We can also observe that the a priori distribution of the 10 classes is (almost) uniform.

Let's implement the LDA:

```
lda.fit<-lda(digit~.,data=train)
lda.fit
```

```
## Call:
## lda(digit ~ ., data = train)
##
## Prior probabilities of groups:
##          0          1          2          3          4          5          6
## 0.10302479 0.10234251 0.10177394 0.09756652 0.10575392 0.09483739 0.09574710
##          7          8          9
## 0.10473050 0.09745281 0.09677053
##
## Group means:
##          x1       y1       x2       y2        x3       y3       x4        y4
## 0 36.114790 86.32230 11.71965 58.85099 14.779249 19.79912 51.06512  7.192053
## 1 14.822222 61.65556 44.43778 78.11222 70.121111 89.61444 77.50333 79.760000
## 2 18.502793 76.94860 42.32626 99.38994 67.776536 79.84469 51.56760 46.111732
## 3 24.811189 83.98368 56.60023 99.50466 86.820513 84.80769 64.46737 60.851981
## 4 42.760215 99.50215 22.06452 79.18495  5.937634 50.98495 43.37527 40.597849
## 5 40.669065 90.95444 42.50480 75.26259 57.489209 58.24820 35.94245 28.601918
## 6 87.697150 98.74228 51.92637 86.88599 20.538005 58.54869  6.71734 26.919240
## 7  3.495114 91.03366 45.19653 98.20521 78.814332 80.99023 71.71987 47.989142
## 8 57.821470 82.11552 39.59277 80.11319 50.931155 52.35589 51.77246 24.417736
## 9 69.126910 81.00705 52.67803 82.68038 45.561692 81.33490 56.73443 83.112808
##         x5        y5       x6        y6       x7        y7        x8        y8
## 0 85.83664 31.247241 89.18543 68.470199 58.85982 89.233996 21.958057 75.097130
## 1 67.84556 54.083333 48.09556 32.643333 44.36333 16.121111 59.444444  1.380000
## 2 20.02458 19.393296 11.50279  9.149721 52.73073  5.241341 98.581006  4.037989
## 3 81.90326 43.488345 90.82751 17.428904 50.11072  2.305361  3.355478  6.203963
## 4 85.44301 49.730108 85.98065 60.110753 70.80323 31.538710 62.301075  0.000000
## 5 25.30695 33.365707 36.64149 51.250600 42.91127 58.804556 60.701439 61.431655
## 6 32.78504  3.119952 81.32185 11.161520 61.21615 30.504751 10.699525 23.059382
```

```
## 7 53.18784 15.567861 34.84691 18.659066 39.95440 33.638436 80.106406 33.878393
## 8 36.33022 16.541424 39.48541 36.654609 66.86231 68.058343 49.992999 81.298716
## 9 79.10458 71.203290 89.19271 43.823737 60.87779 14.920094 18.189189  4.696827
##
## Coefficients of linear discriminants:
##             LD1          LD2          LD3          LD4          LD5
## x1  0.017393141  0.010207297  0.001138530  0.029564905  0.011754422
## y1  0.010508017  0.028825898 -0.020051647 -0.030764165  0.006975022
## x2  0.005990628 -0.001985713 -0.003708872 -0.012976050  0.015755212
## y2 -0.032809227 -0.040448418 -0.008449727 -0.009454702 -0.045377964
## x3 -0.024490509 -0.022968052 -0.010454866  0.019410640 -0.024456588
## y3  0.004508725  0.004960328 -0.020325055  0.010521272  0.038330694
## x4 -0.006247406  0.005728714  0.004298915 -0.008688616  0.013357210
## y4  0.014146445 -0.013277030  0.011534984  0.001292049 -0.010108071
## x5  0.002125993 -0.008430244 -0.022734821 -0.006834414 -0.034832481
## y5 -0.010697263 -0.010490027 -0.031806086  0.012261780  0.021593501
## x6  0.012287031  0.017648186 -0.006482936  0.009566809 -0.006435125
## y6 -0.000405574  0.014092534 -0.063705073  0.005735410  0.020328065
## x7 -0.003019453 -0.004869669  0.007471942  0.001474732 -0.004487174
## y7  0.021943494 -0.007241409  0.057583033 -0.035792758 -0.016183122
## x8 -0.019930813  0.016652426 -0.016642472 -0.009879063  0.012775720
## y8  0.033263032 -0.054147337 -0.051727889  0.019006859  0.014428536
##             LD6          LD7          LD8           LD9
## x1 -0.005157274  0.011450184  0.011095362 -8.176959e-04
## y1 -0.038102461 -0.054052362  0.026644907  1.940819e-02
## x2  0.011225435 -0.026756649  0.004181327  1.174426e-02
## y2 -0.011198038  0.035239698  0.049662118 -7.306762e-02
## x3 -0.019875020  0.017119552  0.010345862 -7.601515e-05
## y3 -0.017162838 -0.033105020 -0.043620939  9.573789e-02
## x4  0.014329760 -0.010668688 -0.001021778  6.309549e-03
## y4  0.054856016 -0.043094272  0.060689477 -5.897154e-02
## x5  0.014787051  0.016544459  0.015728495  2.438254e-02
## y5 -0.030830731  0.026302777 -0.026299041 -3.023683e-02
## x6 -0.019679130 -0.027995303 -0.014460760 -3.010192e-02
## y6 -0.030769115  0.009004666  0.010570340  3.992015e-02
## x7  0.026971201  0.037164279  0.035250716  4.197688e-02
## y7  0.066311266 -0.048492005  0.011132057 -2.296222e-02
## x8 -0.008729406 -0.008753202  0.004730832 -1.846588e-02
## y8 -0.033507365  0.011363514  0.006360155 -4.702182e-03
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5    LD6    LD7    LD8    LD9
## 0.4191 0.1894 0.1224 0.0855 0.0745 0.0494 0.0369 0.0197 0.0030
```

```
lda.fit$prior
```

```
##          0          1          2          3          4          5          6
## 0.10302479 0.10234251 0.10177394 0.09756652 0.10575392 0.09483739 0.09574710
##          7          8          9
## 0.10473050 0.09745281 0.09677053
```

Note that, by default, the priors are the proportions of occurrences of each class in the data set, and they are all pretty similar since as observed the distribution is almost uniform.

We are after the discriminant directions, the coefficient vectors that express the discriminant variables in terms of the original variables. The coefficients are arranged into a 16×9 matrix referred to as Coefficients

of linear discriminants:

```
lda.fit$scaling
```

```
##             LD1          LD2          LD3          LD4          LD5
## x1  0.017393141  0.010207297  0.001138530  0.029564905  0.011754422
## y1  0.010508017  0.028825898 -0.020051647 -0.030764165  0.006975022
## x2  0.005990628 -0.001985713 -0.003708872 -0.012976050  0.015755212
## y2 -0.032809227 -0.040448418 -0.008449727 -0.009454702 -0.045377964
## x3 -0.024490509 -0.022968052 -0.010454866  0.019410640 -0.024456588
## y3  0.004508725  0.004960328 -0.020325055  0.010521272  0.038330694
## x4 -0.006247406  0.005728714  0.004298915 -0.008688616  0.013357210
## y4  0.014146445 -0.013277030  0.011534984  0.001292049 -0.010108071
## x5  0.002125993 -0.008430244 -0.022734821 -0.006834414 -0.034832481
## y5 -0.010697263 -0.010490027 -0.031806086  0.012261780  0.021593501
## x6  0.012287031  0.017648186 -0.006482936  0.009566809 -0.006435125
## y6 -0.000405574  0.014092534 -0.063705073  0.005735410  0.020328065
## x7 -0.003019453 -0.004869669  0.007471942  0.001474732 -0.004487174
## y7  0.021943494 -0.007241409  0.057583033 -0.035792758 -0.016183122
## x8 -0.019930813  0.016652426 -0.016642472 -0.009879063  0.012775720
## y8  0.033263032 -0.054147337 -0.051727889  0.019006859  0.014428536
##             LD6          LD7          LD8           LD9
## x1 -0.005157274  0.011450184  0.011095362 -8.176959e-04
## y1 -0.038102461 -0.054052362  0.026644907  1.940819e-02
## x2  0.011225435 -0.026756649  0.004181327  1.174426e-02
## y2 -0.011198038  0.035239698  0.049662118 -7.306762e-02
## x3 -0.019875020  0.017119552  0.010345862 -7.601515e-05
## y3 -0.017162838 -0.033105020 -0.043620939  9.573789e-02
## x4  0.014329760 -0.010668688 -0.001021778  6.309549e-03
## y4  0.054856016 -0.043094272  0.060689477 -5.897154e-02
## x5  0.014787051  0.016544459  0.015728495  2.438254e-02
## y5 -0.030830731  0.026302777 -0.026299041 -3.023683e-02
## x6 -0.019679130 -0.027995303 -0.014460760 -3.010192e-02
## y6 -0.030769115  0.009004666  0.010570340  3.992015e-02
## x7  0.026971201  0.037164279  0.035250716  4.197688e-02
## y7  0.066311266 -0.048492005  0.011132057 -2.296222e-02
## x8 -0.008729406 -0.008753202  0.004730832 -1.846588e-02
## y8 -0.033507365  0.011363514  0.006360155 -4.702182e-03
```

We can observe that both LD1 and LD2 load heavily on the y-axis elements of the dataset, with a notable emphasis on y_8. This is reasonable because digit doesn't shares their endings, while the central parts are generally similar between most digits, as they typically cross the center of the paper. Moreover, the y-axis elements are more influential than the x-axis ones, as they provide greater variation in the height of the digits, thus potentially serving as a more discriminant factors.

We start using the Fisher derivation: to reproduce the discriminant variables, we first compute the matrix containing the values of the discriminants variables. This is achieved via the matrix product:

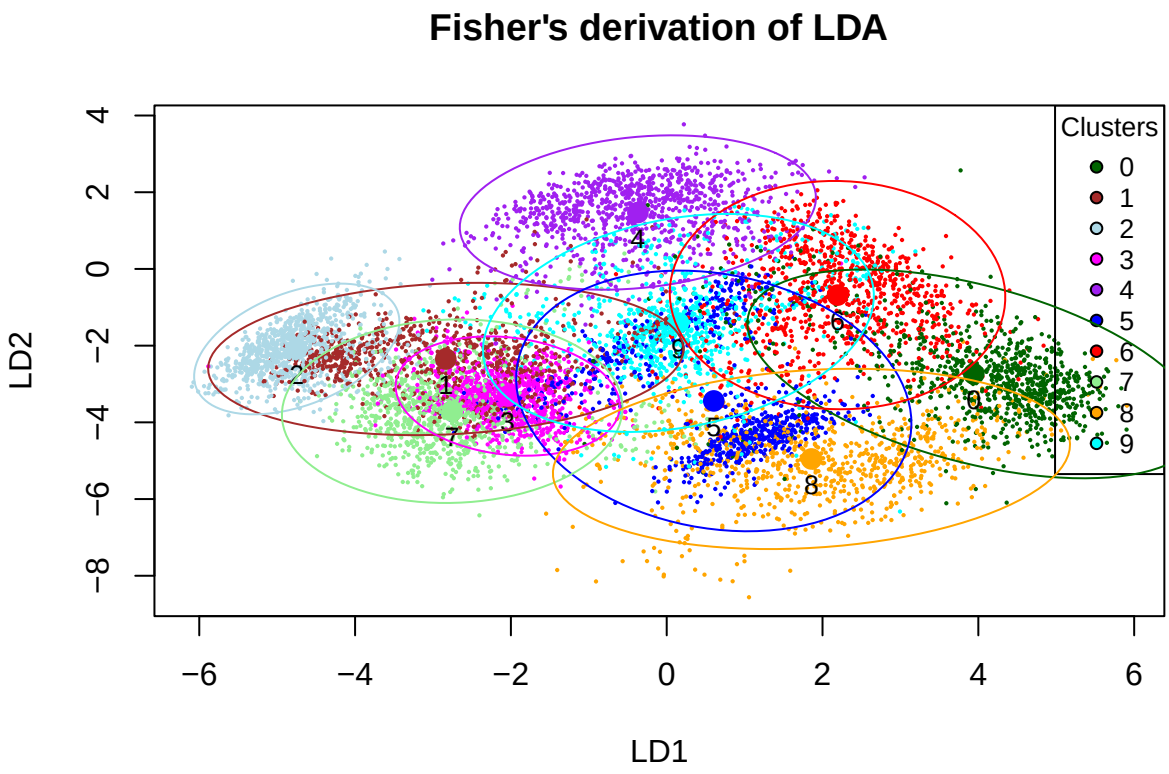
```
A <- lda.fit$scaling
Z <- as.matrix(train[,1:p])%*%A
head(Z)
```

```
##              LD1        LD2        LD3        LD4        LD5        LD6
## 4350  -2.8681603  0.6633356 -11.103706 -2.9614549 -1.0838033 -3.3720119
## 18     0.9589036 -2.0875568 -10.015814  0.9096015 -1.4665033 -3.0393979
## 10903 -2.5749287 -1.8300738 -10.494387 -1.6729427 -1.0359459 -0.3308816
```

```
## 6678  -3.5718643 -3.4708589  -8.461849 -3.9448429 -2.3512038 -2.7839454
## 4788  -0.5978450  0.7838543 -10.827721 -2.3135100 -0.8234077 -3.1595976
## 9481   4.7401519 -4.1451016  -8.953288 -3.6667896 -3.8464345 -2.4980173
##             LD7       LD8         LD9
## 4350  -4.519740  9.957847 -0.27628269
## 18    -4.248081 10.237322 -0.57923671
## 10903 -6.844235 10.015353  0.91860609
## 6678  -6.679239 10.805866  1.45785249
## 4788  -4.875352  9.373597 -0.03615973
## 9481  -5.127793 10.082719 -0.15696479
```

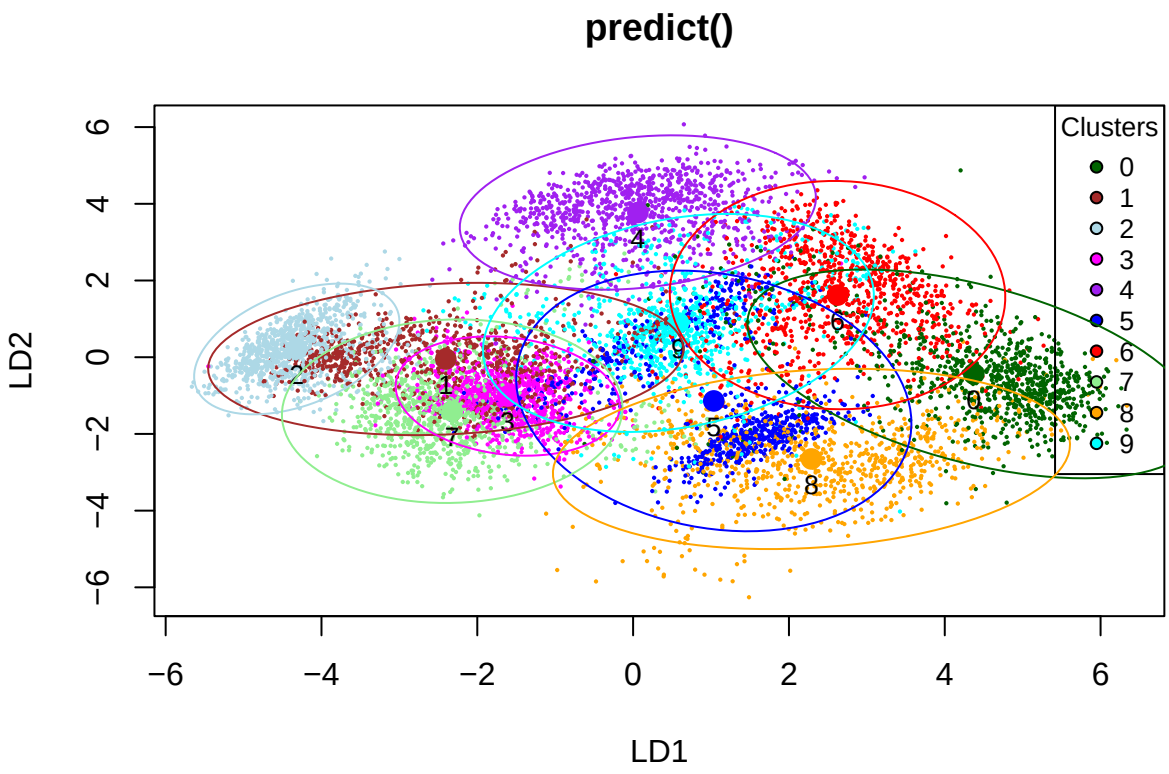
The discriminant variables are ordered by their discriminatory power, making a plot of the first two discriminant variables is informative of the degree of discrimination among classes.

```
labels <- c("0","1","2","3","4","5","6","7","8","9")
centroids <- aggregate(Z[, 1:2], by = list(train$digit), FUN = mean)
centroids <- centroids[, -1]
covariances <- list()
for (i in 1:10) {
  covariances[[i]] <- cov(Z[train[, 17] == (i - 1), 1:2])
}
plot(Z[, 2] ~ Z[, 1], pch = 16, cex=0.3, col = col.index,
     xlab = "LD1", ylab = "LD2", main = "Fisher's derivation of LDA")
points(centroids[, 1], centroids[, 2], cex = 1.5, col = lookup, pch = 16)
legend("topright", legend = labels, pch = 21, pt.bg = lookup,
       title = "Clusters", cex = 0.8)
text(centroids[, 1], centroids[, 2], labels = 0:9, pos = 1, cex = 0.8)
for (i in 1:10) {
  lines(ellipse(x = covariances[[i]][1:2, 1:2], centre = c(t(centroids[i, 1:2])),
                level = 0.95), col = lookup[i])
}
```



We can now achieve the same using the command predict.lda(), which centers the linear discriminant variables so that the weighted mean (weighted by prior) of the group centroids is at the origin.

```
labels <- c("0","1","2","3","4","5","6","7","8","9")
pred = predict(lda.fit)
centroids = aggregate(pred$x, by = list(train$digit), FUN = mean)
centroids = centroids[, -1]
covariances = list()
for (i in 1:10){
covariances[[i]] = cov(pred$x[train[, 17] == (i - 1), ])
}

plot(LD2 ~ LD1, data = pred$x, pch = 16, cex=0.3, col = col.index,
xlab = "LD1", ylab = "LD2", main = "predict()")
points(centroids[, 1], centroids[, 2], cex = 1.5, col = lookup, pch = 16)

legend("topright", legend = labels, pch = 21, pt.bg = lookup,
       title = "Clusters", cex = 0.8)

text(centroids[, 1], centroids[, 2], labels = 0:9, pos = 1, cex = 0.8)

for (i in 1:10) {
  lines(ellipse(x = covariances[[i]][1:2, 1:2], centre = c(t(centroids[i, 1:2])),
                level = 0.95), col = lookup[i])
}
```



As expected they don't produce the same result.

```
head(Z)
```

```
##              LD1        LD2        LD3        LD4        LD5        LD6
## 4350  -2.8681603  0.6633356 -11.103706 -2.9614549 -1.0838033 -3.3720119
## 18     0.9589036 -2.0875568 -10.015814  0.9096015 -1.4665033 -3.0393979
## 10903 -2.5749287 -1.8300738 -10.494387 -1.6729427 -1.0359459 -0.3308816
## 6678  -3.5718643 -3.4708589  -8.461849 -3.9448429 -2.3512038 -2.7839454
## 4788  -0.5978450  0.7838543 -10.827721 -2.3135100 -0.8234077 -3.1595976
## 9481   4.7401519 -4.1451016  -8.953288 -3.6667896 -3.8464345 -2.4980173
##              LD7       LD8         LD9
## 4350  -4.519740  9.957847 -0.27628269
## 18    -4.248081 10.237322 -0.57923671
## 10903 -6.844235 10.015353  0.91860609
## 6678  -6.679239 10.805866  1.45785249
## 4788  -4.875352  9.373597 -0.03615973
## 9481  -5.127793 10.082719 -0.15696479
```

```
head(pred$x)
```

```
##              LD1        LD2         LD3        LD4        LD5        LD6
## 4350  -2.4369512  2.9650202 -2.61065403 -1.1559062  0.2252161 -0.4638188
## 18     1.3901127  0.2141279 -1.52276150  2.7151502 -0.1574838 -0.1312048
## 10903 -2.1437196  0.4716109 -2.00133413  0.1326060  0.2730735  2.5773115
## 6678  -3.1406552 -1.1691743  0.03120369 -2.1392942 -1.0421844  0.1242478
## 4788  -0.1666359  3.0855389 -2.33466878 -0.5079613  0.4856117 -0.2514044
## 9481   5.1713610 -1.8434170 -0.46023532 -1.8612409 -2.5374150  0.4101758
```

```
##              LD7        LD8        LD9
## 4350   0.8514881  0.3832142 -0.6631788
## 18     1.1231470  0.6626899 -0.9661328
## 10903 -1.4730070  0.4407208  0.5317100
## 6678  -1.3080111  1.2312337  1.0709564
## 4788   0.4958759 -0.2010352 -0.4230558
## 9481   0.2434355  0.5080864 -0.5438609
```

As anticipated, the second plot appears to be more centered around the origin. This alignment is a result of centering the linear discriminants so that the weighted mean, which is essentially the arithmetic mean of the centroids due to nearly equal prior probabilities, coincides with the origin. Consequently, the grand mean vector is also situated at the origin. This adjustment facilitates a simpler interpretation of the decision boundaries. For further ease of interpretation, we utilize the predict() function in our analysis.

To understand the overlap between different clusters we have added the countor plot of a gaussian distribution with means the centroids and covariance matrix the sample covariance matrix at level of 0.95, in the plots for each of the 10 classes. First of all, we notice that the first two Linear Discriminant directions capture only around 60% of the data variance. Digits 2 and 4 form well-separated clusters, located on the left and top of the plot, respectively. Digit 5 seems to be spread in two parts, while others show varying degrees of spread around their centroids. Overlap is evident between one region of 5 and the group 9 and from the other region of 5 and 8. For this reason, we will expect mis-classifications of 5. Also, 3, 7, and 1 are overlapping. The cluster 0 seems to be well separated from the others but has a partial overlap with 6.

2.2

Compute the confusion matrix on the training data. What are the groups more difficult to separate from the others? Comment in view of the answer to point 1.
We computed the confusion matrix

```
lda.pred<-predict(lda.fit)
predicted <- lda.pred$class
```

```
conf.mat <- table(predicted=predicted,real=train$digit)
conf.mat
```

```
##          real
## predicted   0   1   2   3   4   5   6   7   8   9
##         0 801   0   0   0   0   0   6   1  67   1
##         1   6 625  20  16   1   1   0  51  15  50
##         2   0 163 867   1   2   0   0  12   0   0
##         3   0  15   1 827   1  41   0  16   9   9
##         4  30   2   0   0 905   0   1   5   0  16
##         5   0  32   0   0   0 575   2   5  42   9
##         6   3   5   0   0   3   2 822   0   4   0
##         7   0  25   7  13   1   1   0 817   1   0
##         8  64   0   0   0   0   4  11   4 709   2
##         9   2  33   0   1  17 210   0  10  10 764
```

and the full rank LDA training error:

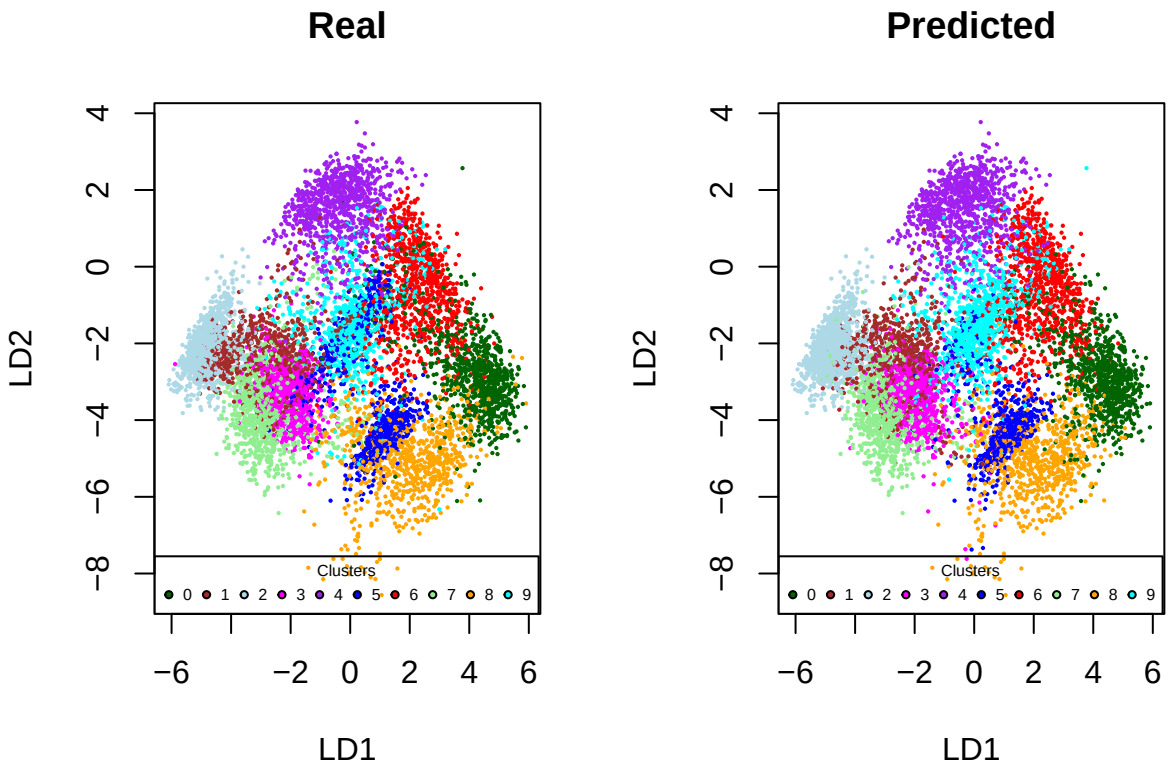
```
AER <- 1-mean(lda.pred$class==train$digit)
print(AER)
```

```
## [1] 0.1230384
```

The training error rate is approximately 0.12, indicating that on the test set, the classification is accurate

nearly 9 times out of 10. In the next image we plotted the data according to the values of the first two linear discriminant variables and color coded them according to their class, in this we we show a comparison of the real and predicted class.

```
labels <- c("0","1","2","3","4","5","6","7","8","9")
par(mfrow=c(1,2))
col.index.pred <- lookup[as.numeric(predicted)]
plot(Z[,1],Z[,2],col=col.index,pch=16, cex=0.3,
     xlab = "LD1", ylab = "LD2", main = "Real")
legend(x = "bottomleft", y = "bottom", legend = labels, pch = 21, pt.bg = lookup,
       title = "Clusters", cex = 0.5, horiz = TRUE)
plot(Z[,1],Z[,2],col=col.index.pred,pch=16, cex=0.3,
     xlab = "LD1", ylab = "LD2", main = "Predicted")
legend(x = "bottomleft", y = "bottom", legend = labels, pch = 21, pt.bg = lookup,
       title = "Clusters", cex = 0.5, horiz = TRUE)
```



To determine which group is harder to classify, we consider two parameters: the misclassification rate and the "posterior misclassification probability". While the misclassification rate indicates the proportion of correct predictions among all values, the misclassification probability represents the likelihood that a predicted value actually belongs to the classified set. This distinction is crucial, as it illustrates the potential limitations of a simplistic decision rule. For instance, if tasked with distinguishing between cats and dogs, consistently labeling everything as a dog yields a perfect accuracy in identifying dogs, but this alone is not a realiable information to evaluate the model. Thus, the misclassification rate is the main parameter under which we base our analysis but we can also observe the misclassification probability since it offers additional insights.

```
par(mfrow=c(1,1))
mis.prob <- (rowSums(conf.mat)-diag(conf.mat))/(rowSums(conf.mat))
```

```r
cat("Posteriori misspecification probability per class:\n")
```

```
## Posteriori misspecification probability per class:
```

```r
print(sort(round(mis.prob,3), decreasing = F))
```

```
##     6     7     4     0     3     8     5     2     1     9
## 0.020 0.055 0.056 0.086 0.100 0.107 0.135 0.170 0.204 0.270
```

```r
total_misclassification <- colSums(conf.mat) - diag(conf.mat)
misclassification_rate <- round(total_misclassification / lda.fit$counts, 4)
cat("Total misclassifications per class:\n")
```

```
## Total misclassifications per class:
```

```r
print(total_misclassification)
```

```
##   0   1   2   3   4   5   6   7   8   9
## 105 275  28  31  25 259  20 104 148  87
```

```r
cat("\nMisclassification rate per class:\n")
```

```
##
## Misclassification rate per class:
```

```r
print(misclassification_rate)
```

```
##      0      1      2      3      4      5      6      7      8      9
## 0.1159 0.3056 0.0313 0.0361 0.0269 0.3106 0.0238 0.1129 0.1727 0.1022
```

Digits 1 and 5 exhibit the highest misclassification rates, with approximately one-third of their observations being misclassified. These digits also frequently overlapped with other classes in the scatterplot. In contrast, digits 2, 4, and 6 demonstrate excellent classification results, this was also evident from the previous scatterplot since the corresponding clusters where highly concentrated around their centroids. Additionally, digits 8, 0, 7, and 9 present a misclassification rate of 10%. Particularly, 8, 0, and 7 also have a high probability of being in the corrected class given the fact that they are predicted as 8, 0, and 7. Digit 9 present a misclassification rate that isn't excessively high. However, the posterior probability reveals that nearly all items inaccurately classified actually fall into the category 9, with the digit 5 being the primary contributor to this misclassification. Therefore, when a digit is classified as 9, it introduces a greater level of uncertainty regarding the precision of our classifications. Notably, digit 0 is frequently misclassified as 8. Some digits exhibit a bias in their misclassification direction, such as digit 1 being misclassified as 2 more frequently than then contrary. This discrepancy is explained by the high variance of digit 1, leading to overlaps with digit 2 in the scatterplot. Despite observed misclassifications, the overall training error rate is low at 12,3% indicating that on the train set, the classification is accurate nearly 9 times out of 10. As expected, there are no misclassifications between the most distant classes in the LD1-LD2 space such as 2 and 0.

2.3

Use leave-one-out cross validation (CV). Compute the confusion matrix and the corresponding CV error. Is it larger than the training error? Why so?

In leave-one-out cross-validation, the model is trained iteratively while excluding one observation at a time, reserving it solely for testing purposes. This method allows us to assess the model's performance on new data by simulating its behavior with unseen instances. As we begin the leave-one-out cross-validation process, we compute the subsequent confusion matrix.

```r
lda.fitCV<-lda(digit~.,data=pendigits,CV=T)
conf.matCV<-table(predicted=lda.fitCV$class,true=pendigits$digit)
conf.matCV
```

```
##           true
## predicted    0    1    2    3    4    5    6    7    8    9
##         0 1013    0    0    0    0    0    4    1   79    1
##         1   10  798   23   19    1    3    0   61   18   65
##         2    0  206 1113    1    2    0    0   17    0    0
##         3    0   16    1 1020    1   58    0   20   10   10
##         4   34    2    0    0 1115    0    3    7    0   20
##         5    0   51    0    0    0  712    5    5   57   13
##         6    5    8    0    0    4    3 1029    0    7    1
##         7    0   29    7   13    1    0    0 1013    2    0
##         8   78    0    0    0    0    9   15    6  871    2
##         9    3   33    0    2   20  270    0   12   11  943
```

```
ERCV<-1-mean(lda.fitCV$class==pendigits$digit)
cat("The training error is: ", AER, "\n")
```

```
## The training error is:  0.1230384
```

```
cat("The CV error is: ", ERCV, "\n")
```

```
## The CV error is:  0.1241812
```

Now, we calculate the error in the test dataset (previously created).

```
lda.pred.test<-predict(lda.fit,test)
predicted.test <- lda.pred.test$class
conf.mat.test <- table(predicted=predicted.test,real=test$digit)
AER.test <- mean(lda.pred.test$class!=test$digit)
cat("The test error is: ", AER.test, "\n")
```

```
## The test error is:  0.1242038
```

The training error naturally tends to be lower than both the test error and the CV error. This depends on the fact that the classifier is being assessed on the same data it was trained on, potentially leading to an underestimation of the true error rate on unseen data. Therefore the test error, calculated on data on which the model was not trained, prevents the risk of underestimation, and is a more realistic indicator of the accuracy of the model.

Leave-one-out cross-validation also provides a more realistic evaluation. Moreover, with this method, the model is trained on all but one data point and then tested on the omitted point, thus this approach minimizes the "loss" of data in the training procedure since only one element is excluded. A small discrepancy between the CV error and the training error can be interpreted positively, suggesting that the model avoids overfitting the training data.

The slight difference between the CV error and the test error could be attributed to the fact that more data are used for training the model in cross-validation than for assessing the error. However, the closeness of the three results indicates a good fit of the model to our data.

2.4

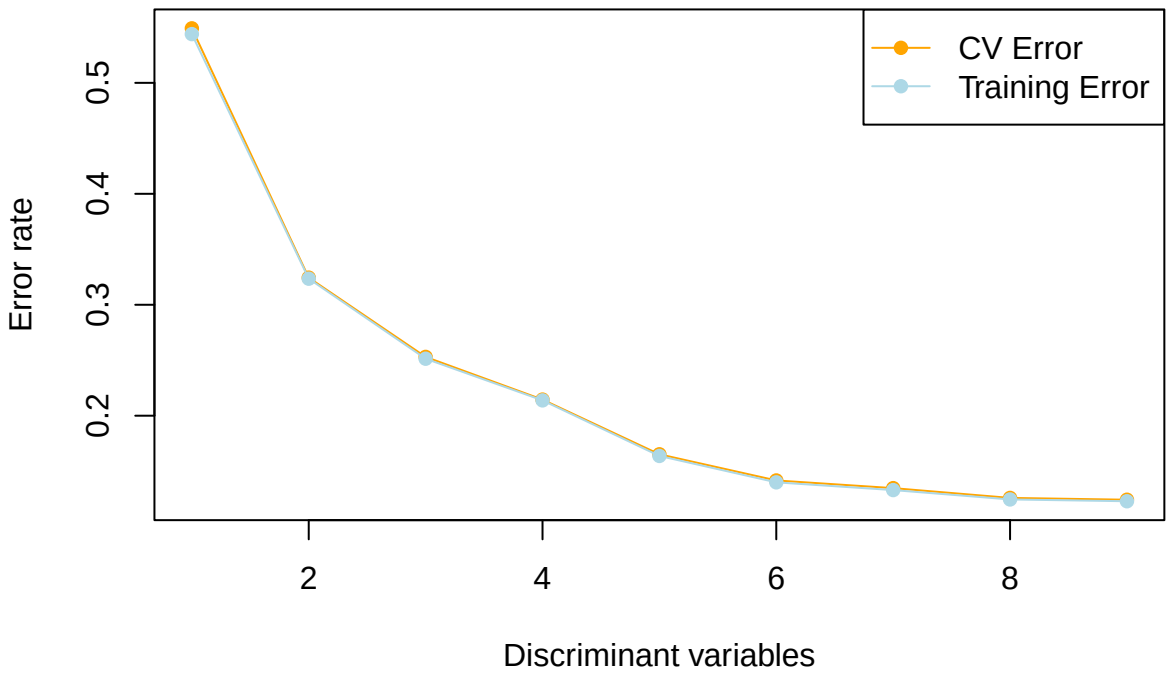
Compute the 44-fold cross validation error for each reduced-rank LDA classifier, including full-rank LDA, by using the partition of the observations provided by the variable groupCV below. Plot the error curve against the number of discriminant variables. What classifier do you prefer? Comment.

A 44-fold cross-validation procedure is used to evaluate a model's performance in classifying handwritten digits. The process involves dividing the data into 44 groups. We train a model 44 times; each time we use as training sets 43 groups, and then evaluate the model on the remaining gropu which serves as test set. A 44-fold cross-validation could potentially effectively assess our model's predictive capability if observations

were grouped by "author," given there are 44 different authors (but we don't have this information). We repeat this cross-validation procedure using reduced-rank linear discriminant analysis, which allows nearest centroid classification in a subspace of dimension $K < 9$ by utilizing only the first K discriminant directions for prediction. While performing the 44-fold cross-validation, the final fold does contain eight data points less compared to the others. Theoretically, this difference should be addressed to ensure a more accurate error estimation. However, we consider this a negligible issue in this case. With only eight observations out of a total of approximately 250, the size difference translates to a fraction around 0.032. This suggests minimal impact on the overall error estimate.

```
groupCV<-rep(1:44, each=250)
groupCV<-groupCV[1:length(pendigits$digit)]
ERCVk.vec <- rep(0,9)
ERCVk.vec.train <- rep(0,9)
for (k in 1:9) {
  ERCV.vec <- rep(0,44)
  ERCV.vec.train <- rep(0,44)
  for (i in 1:44) {
    test.i <- pendigits[groupCV==i,]
    train.i <- pendigits[!(groupCV==i),]
    lda.fit.i <- lda(digit~.,data=train.i)
    predict.i <- predict(lda.fit.i,test.i,dimen=k)
    predict.i.train <- predict(lda.fit.i,dimen=k)
    ERCV.i <- mean(predict.i$class!=test.i$digit)
    ERCV.vec[i]=ERCV.i
    ERCV.i.train <- mean(predict.i.train$class!=train.i$digit)
    ERCV.vec.train[i]=ERCV.i.train
  }
  ERCV.k <- mean(ERCV.vec)
  ERCVk.vec[k]=ERCV.k
  ERCV.k.train <- mean(ERCV.vec.train)
  ERCVk.vec.train[k]=ERCV.k.train
}
```

```
plot(ERCVk.vec, xlab = "Discriminant variables", ylab = "Error rate", type = "o",
     col = "orange", pch = 16, ylim = range(ERCVk.vec, ERCVk.vec.train))
points(ERCVk.vec.train, type = "o", col = "lightblue", pch = 16)
legend("topright", legend = c("CV Error", "Training Error"),
       col = c("orange", "lightblue"), pch = 16, lty = 1)
```




```
table <- matrix(c(ERCVk.vec, ERCVk.vec.train), ncol = 9)
rownames(table) <- c("CV error", "Training Error")
colnames(table) <- 1:9
print(table)
```

```
##                       1         2         3         4         5         6
## CV error       0.5491300 0.2530075 0.1652261 0.1347355 0.1242810 0.3234974
## Training Error 0.3244313 0.2145289 0.1416476 0.1259144 0.5439474 0.2513710
##                       7         8         9
## CV error       0.2138342 0.1399517 0.1245303
## Training Error 0.1637407 0.1329677 0.1229118
```

As expected, the CV error remains slightly higher than the training error. This discrepancy arises because cross-validation offers a more reliable estimate of the model's generalization performance, assessing its efficacy on unseen data.

Additionally, we note a decline in the CV error as the number of discriminant variables increases. This trend implies that incorporating more variables increases the model's capacity to distinguish between classes, then the best choice is to incorporate all 9 discriminant variable.

2.5

(Optional) Find a classification rule that improves on the CV error rate estimates found before. Feel free to use any classification method, even one not covered in class.
In point 1 we saw that the hypothesis on the omoschedasticity of the groups was not verified, so to get an improvement in the classification, the first idea that comes to mind is to use a quadratic discriminant analysis, which does not require the covariance matrices of the groups to be equal.

However we have to notice that QDA requires the covariance matrices to be invertible, but as we already observed, the observations in group 4 have one variable (y_8) which is constantly null, hence the covariance matrix is not full rank and cannot be inverted.

```
head(train[train$digit==4,1:16])
```

```
##       x1  y1 x2  y2 x3 y3  x4 y4  x5 y5  x6 y6 x7 y7  x8 y8
## 4350  11 100  0  74 43 56 100 56  88 83  87 55 84 27  87  0
## 4788  46 100  0  75 14 57  85 57 100 82  74 63 53 31  46  0
## 9994  63 100 24  88  0 73   4 57  41 45  76 32 92 16 100  0
## 3335  20  92  5 100  0 76   2 52  52 33  95 47 80 23 100  0
## 7529 100  95 75 100 42 70   0 44  42 33  85 49 83 31  85  0
## 6797  33 100 21  70  0 37  29 27  83 35 100 62 88 35  80  0
```

```
cov(train[train$digit==4,1:16])
```

```
##             x1         y1          x2          y2         x3          y3
## x1  590.787436  5.0204301  327.041321  -25.781224  21.062532   -4.920729
## y1    5.020430  7.9079609   -8.889267   -8.777578  -5.066604   -7.557556
## x2  327.041321 -8.8892670  368.542658   37.073093  45.072919   46.718949
## y2  -25.781224 -8.7775779   37.073093   85.384488   7.851159   86.467803
## x3   21.062532 -5.0666042   45.072919    7.851159 125.472963   31.528662
## y3   -4.920729 -7.5575564   46.718949   86.467803  31.528662  120.103110
## x4 -133.849639 10.6563503 -352.904754 -103.858500  17.433547 -127.317273
## y4   -3.994273 -1.5825573  -25.337859   28.945285  35.049166   59.805565
## x5  -59.136931  7.2789171 -142.994166  -69.502904 -75.284505 -111.593970
## y5  -46.465216  2.6469750 -150.327025  -12.007081  30.576261   -1.093411
## x6    4.733782 -1.7502274   91.200389   12.909934 -31.824674    2.262356
## y6    4.049190 -0.7758036  -69.953332  -32.335898 -16.866067  -76.628040
## x7  -77.060151  0.3594118   39.618737    8.600479 -27.819605   -2.091232
## y7   12.345675 -1.7282857  -12.510573  -17.289191  -3.005118  -42.276058
## x8 -176.676922  2.7463917  -11.570575   14.020684 -36.775606   -3.672535
## y8    0.000000  0.0000000    0.000000    0.000000   0.000000    0.000000
##            x4         y4          x5          y5          x6          y6
## x1 -133.84964  -3.994273  -59.136931  -46.465216    4.733782   4.0491904
## y1   10.65635  -1.582557    7.278917    2.646975   -1.750227  -0.7758036
## x2 -352.90475 -25.337859 -142.994166 -150.327025   91.200389 -69.9533317
## y2 -103.85850  28.945285  -69.502904  -12.007081   12.909934 -32.3358982
## x3   17.43355  35.049166  -75.284505   30.576261  -31.824674 -16.8660671
## y3 -127.31727  59.805565 -111.593970   -1.093411    2.262356 -76.6280403
## x4 1005.45106 140.936868  415.900311  471.843049 -279.570770 339.0660359
## y4  140.93687 116.023246    8.062095  137.964543  -77.565381  22.3955022
## x5  415.90031   8.062095  420.442927  175.054032  -30.431675 179.4417329
## y5  471.84305 137.964543  175.054032  327.170352 -146.974013 153.8146382
## x6 -279.57077 -77.565381  -30.431675 -146.974013  221.429119 -83.8094795
## y6  339.06604  22.395502  179.441733  153.814638  -83.809479 240.4387432
## x7 -265.54179 -72.969423 -122.046210 -157.817428  170.521483 -85.0965902
## y7  108.31646  -8.167409   57.499517   34.390975  -24.674169  93.3030279
## x8 -297.26273 -88.997199 -178.290681 -194.408429  154.363207 -91.6103453
## y8    0.00000   0.000000    0.000000    0.000000    0.000000   0.0000000
##              x7         y7          x8 y8
## x1  -77.0601514  12.345675 -176.676922  0
## y1    0.3594118  -1.728286    2.746392  0
```

```
## x2   39.6187368 -12.510573  -11.570575  0
## y2    8.6004792 -17.289191   14.020684  0
## x3  -27.8196048  -3.005118  -36.775606  0
## y3   -2.0912323 -42.276058   -3.672535  0
## x4 -265.5417931 108.316459 -297.262729  0
## y4  -72.9694225  -8.167409  -88.997199  0
## x5 -122.0462099  57.499517 -178.290681  0
## y5 -157.8174277  34.390975 -194.408429  0
## x6  170.5214834 -24.674169  154.363207  0
## y6  -85.0965902  93.303028  -91.610345  0
## x7  310.8837355 -25.762558  408.564152  0
## y7  -25.7625577  50.263839  -29.631689  0
## x8  408.5641515 -29.631689  658.953390  0
## y8    0.0000000   0.000000    0.000000  0
```

One possible solution to this problem could be using principal component analysis on the data to provide a reduction of the dimension which should eliminate collinearity problems. As we already discussed in the previous problem set, we choose 5 components, and redefine the train and test sets on the data with reduced dimensionality.

```
PCA <- prcomp(pendigits[1:16],scale=T)
summary(PCA)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.1718 1.7970 1.6052 1.10894 1.03107 0.89294 0.77888
## Proportion of Variance 0.2948 0.2018 0.1610 0.07686 0.06644 0.04983 0.03792
## Cumulative Proportion  0.2948 0.4966 0.6577 0.73452 0.80096 0.85079 0.88871
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.74044 0.64096 0.54611 0.45882 0.33511 0.28369 0.24084
## Proportion of Variance 0.03427 0.02568 0.01864 0.01316 0.00702 0.00503 0.00363
## Cumulative Proportion  0.92297 0.94865 0.96729 0.98045 0.98747 0.99250 0.99612
##                           PC15    PC16
## Standard deviation     0.18511 0.16673
## Proportion of Variance 0.00214 0.00174
## Cumulative Proportion  0.99826 1.00000
```

```
reduced_train <- (PCA$x)[ind,]
reduced_train <- as.data.frame(reduced_train[,1:5])
reduced_train$digit <- train$digit
reduced_test <- (PCA$x)[-ind,]
reduced_test <- as.data.frame(reduced_test[,1:5])
reduced_test$digit <- test$digit
```

Now we can implement again QDA and see how the new model classifies the data in the test set.

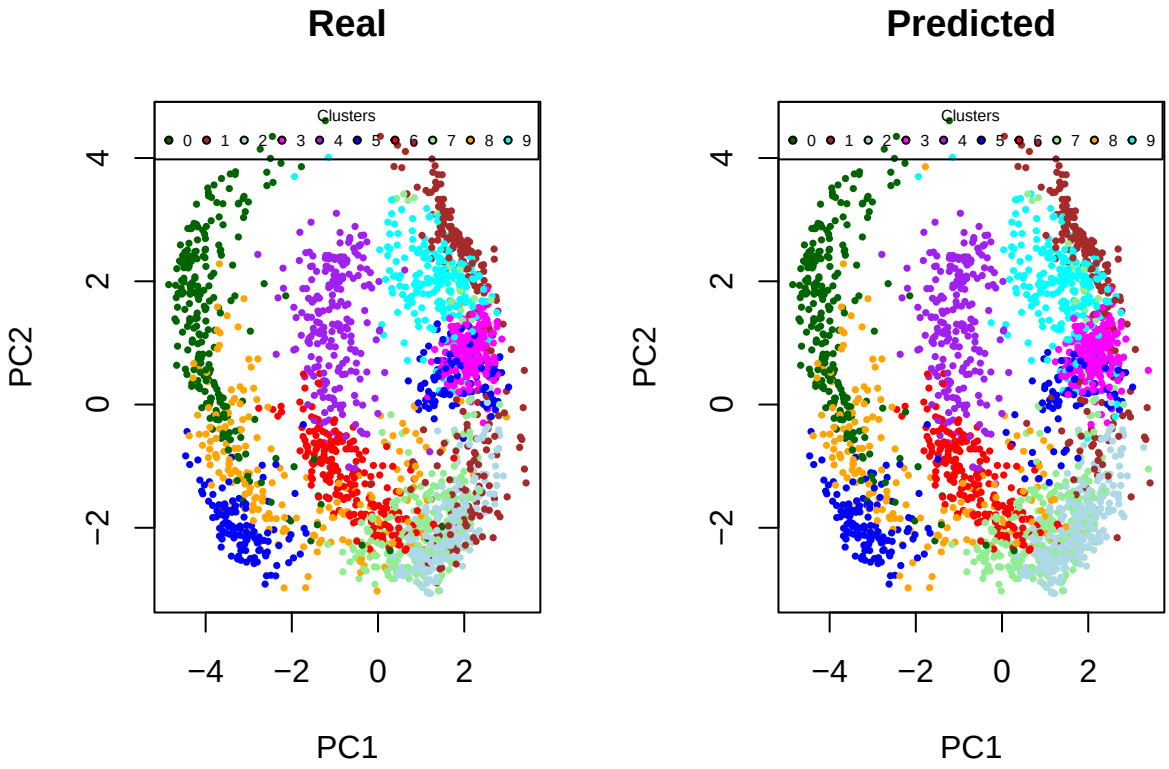
```
qda.fit <- qda(digit~.,data=reduced_train)
qda.pred.test<-predict(qda.fit,reduced_test)
qda.predicted.test <- qda.pred.test$class
QDA.ER <- mean(qda.pred.test$class!=test$digit)
QDA.ER
```

```
## [1] 0.09008189
```

We get a slight improvement in the classification error.

Next we plot the data (using the first two principal components instead of the linear discriminant variables) comparing the real and predicted.

```
par(mfrow=c(1,2))
col.index <- lookup[(reduced_test$digit)+1]
col.index.pred <- lookup[as.numeric(qda.predicted.test)]
plot(reduced_test[,1],reduced_test[,2],col=col.index,pch=16,cex = 0.5,xlab="PC1",ylab="PC2",main="Real")
legend(x = "topleft", y = "bottom", legend = labels, pch = 21, pt.bg = lookup,
       title = "Clusters", cex = 0.5, horiz = TRUE)
plot(reduced_test[,1],reduced_test[,2],col=col.index.pred,pch=16,cex = 0.5,xlab="PC1",ylab="PC2",main="
legend(x = "topleft", y = "bottom", legend = labels, pch = 21, pt.bg = lookup,
       title = "Clusters", cex = 0.5, horiz = TRUE)
```




```
par(mfrow=c(1,1))
```

As an alternative approach we can implement the random forest classification. Here is how it works in summary:

- the dataset is split in random subset with replacement (meaning that we are not doing a partition);
- a random tree is built for each subset using a random selection of predictor variables (in this way the trees are less correlated), which determines at each node of the tree the splitting of the original subset of data.
- the splittings are made in order to obtain "purer" nodes, that is nodes in which the majority of observations belong to the same class. There are many possible criteria, one of the most intuitive is to choose the split that minimizes the Gini impurity index, which is defined by

$$G(t) = 1 - \sum_{i=1}^{C} p(i|t)^2$$

where $p(i|t)$ is the relative frequency of the i-th class i n the node t;

- the predictions of all trees are aggregated by a majority vote.

The greatest advantage of a random forest is that it avoids overfitting since many decision trees are used simultaneously.

```
train$digit <- factor(train$digit)
rf <- randomForest(digit ~ ., data = train, type="classification")
rf_pred <- predict(rf, test)
RFER <- mean(rf_pred!=test$digit)
RFER
```

```
## [1] 0.00955414
```

```
confrandom<- table(predicted=rf_pred,real=test$digit)
confrandom
```

```
##          real
## predicted   0   1   2   3   4   5   6   7   8   9
##         0 236   0   0   0   0   0   0   0   0   0
##         1   0 234   2   0   0   0   0   0   0   0
##         2   0   9 247   0   0   0   0   0   0   0
##         3   0   0   0 193   0   2   0   0   0   0
##         4   0   0   0   0 214   0   1   0   0   0
##         5   0   0   0   0   0 219   1   0   0   0
##         6   0   0   0   0   0   0 212   0   0   0
##         7   0   0   0   3   0   0   0 221   0   1
##         8   0   0   0   0   0   0   0   0 198   0
##         9   1   0   0   1   0   0   0   0   0 203
```

As we see, the error rate decreases significantly, as the model produces only 1 mis-classification every 100. In particular, the misspecification rate, decreases towards zero