Exercise 2 - MSA

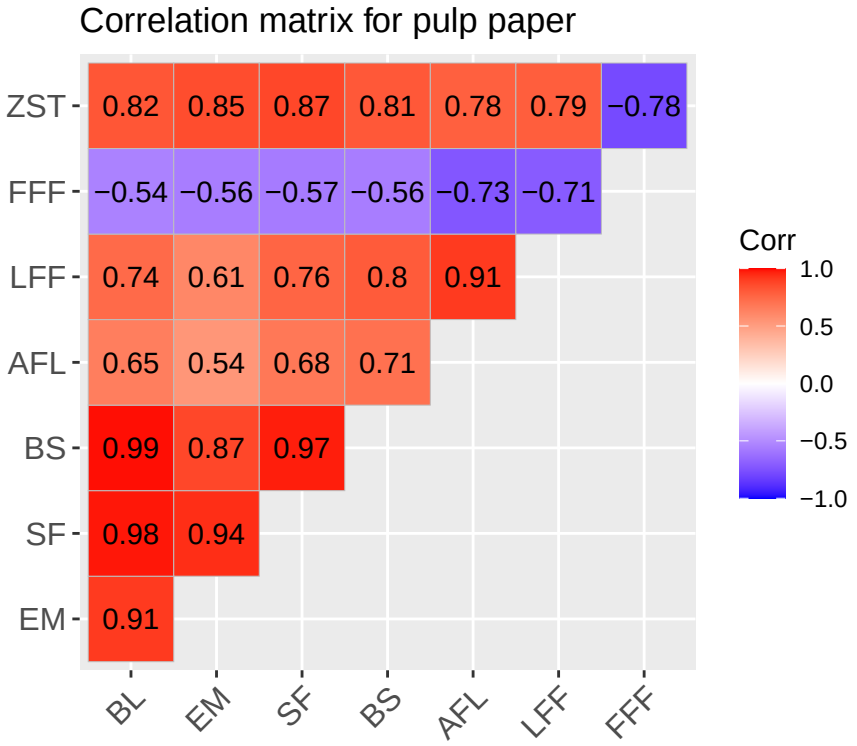
Matteo Bandiera - Samuele Fonio - Luca Macis

Exercise 1

Point 1.1

First of all we show the summary of our data set and the correlation matrix on the standardized observations, in order to see if a Factor Analysis can be performed with useful results.

BL	EM	SF	BS	AFL	LFF	FFF	ZST
Min. :16.29	Min. :5.948	Min. :2.997	Min. :-0.4780	Min. :-0.69400	Min. : 1.515	Min. :-0.391	Min. :0.998
1st Qu.:19.33	1st Qu.:6.579	1st Qu.:4.364	1st Qu.: 0.4602	1st Qu.:-0.15400	1st Qu.:31.913	1st Qu.:17.987	1st Qu.:1.044
Median :21.26	Median :7.386	Median :5.357	Median : 0.9260	Median : 0.00000	Median :41.925	Median :27.085	Median :1.066
Mean :21.72	Mean :7.266	Mean :5.637	Mean : 1.0188	Mean :-0.02176	Mean :39.033	Mean :26.678	Mean :1.067
3rd Qu.:24.52	3rd Qu.:7.913	3rd Qu.:7.102	3rd Qu.: 1.7025	3rd Qu.: 0.13100	3rd Qu.:49.025	3rd Qu.:33.775	3rd Qu.:1.088
Max. :26.19	Max. :8.454	Max. :7.816	Max. : 2.1450	Max. : 0.55800	Max. :63.035	Max. :84.554	Max. :1.129

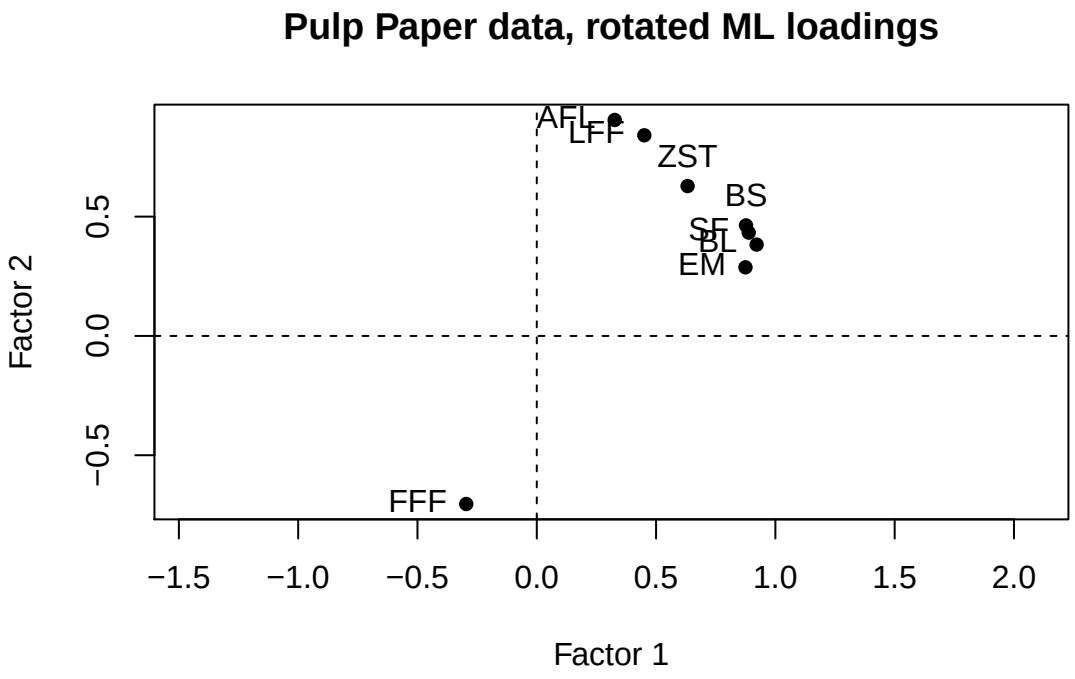


As we can see from the correlation matrix there is a strong correlation between the variables, so it is useful performing a Factor analysis.
Let's start by m=2.
For performing the maximum likelihood method we used factanal() and, since without rotation the loadings were not clear enough, for the belonging to the two factors, we used varimax for getting the best division in

the two factors:

	ML1	ML2	h2	u2
BL	0.921	0.383	0.995	0.005
EM	0.875	0.288	0.848	0.152
SF	0.888	0.433	0.977	0.023
BS	0.877	0.463	0.984	0.016
AFL	0.327	0.905	0.926	0.074
LFF	0.451	0.841	0.910	0.090
FFF	-0.296	-0.704	0.584	0.416
ZST	0.632	0.628	0.794	0.206

Note: h2 are the communalities and u2 are the specific variances.
We can see that BL, EM, SF, BS load really high on factor 1. AFL, LFF now load higher on factor 2. FFF loads negative on both factors (-0.30, -0.70). ZST loads the same on both (0.63, 0.63). In general we can recognize two groups: one composed by the first four variables, and the second from the others. Note that in text of the exercise there was already this separation, so we confirm it. However ZST has a strange behavior, but it will be investigated later on in the second point. If we want to visualize the loadings:



As expected the first group loads high on the factor 1, ZST is in the middle and the other (in absolute value) load high on the second factor.

We now see the residual matrix, which we remember is a matrix showing the difference between the approximated correlation matrix and the real correlation matrix.

	BL	EM	SF	BS	AFL	LFF	FFF	ZST
BL	0.000	-0.002	0.000	0.002	0.000	-0.002	0.000	-0.001
EM	-0.002	0.000	0.041	-0.026	-0.009	-0.027	-0.095	0.116
SF	0.000	0.041	0.000	-0.005	-0.001	0.000	-0.007	0.032
BS	0.002	-0.026	-0.005	0.000	0.001	0.011	0.022	-0.032
AFL	0.000	-0.009	-0.001	0.001	0.000	-0.003	0.001	0.009
LFF	-0.002	-0.027	0.000	0.011	-0.003	0.000	0.015	-0.020
FFF	0.000	-0.095	-0.007	0.022	0.001	0.015	0.000	-0.155
ZST	-0.001	0.116	0.032	-0.032	0.009	-0.020	-0.155	0.000

And of course the residual error using 2 factors:

Sum squared of residual
0.1064641

As we can see the error is not so small, but with two factor the reduction is important. Furthermore note that there is ZST in which we noticed a strange behavior.

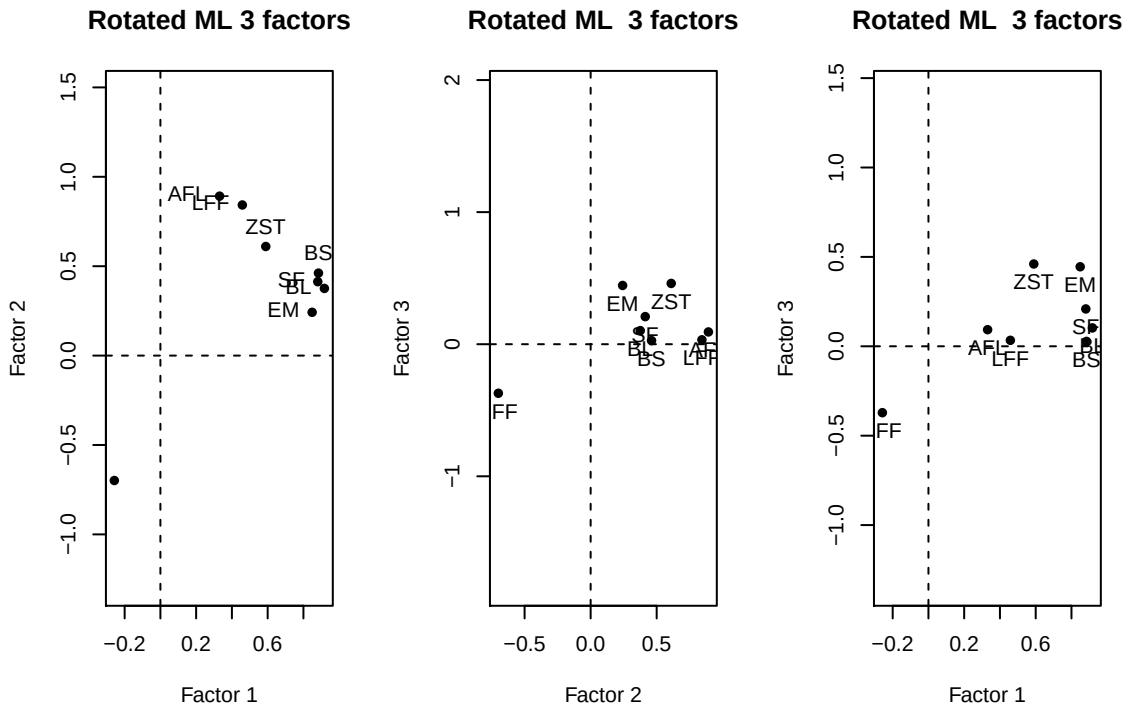
To conclude the requested outputs here is the proportion of total variance and the cumulative proportion of total variance:

	Proportion of variance	Cumulative proportion
Factor1	0.4961235	0.4961235
Factor2	0.3810058	0.8771294

From this we can see that with two factors we can explain the 88% of the variance, that is a satisfactory result.
Now we can start the analysis for three factors, so we can compare the results.

	ML1	ML2	ML3	h2	u2
BL	0.917	0.376	0.103	0.993	0.007
EM	0.849	0.242	0.445	0.977	0.023
SF	0.880	0.413	0.209	0.989	0.011
BS	0.884	0.461	0.027	0.995	0.005
AFL	0.331	0.892	0.093	0.913	0.087
LFF	0.458	0.842	0.033	0.921	0.079
FFF	-0.258	-0.699	-0.371	0.693	0.307
ZST	0.589	0.610	0.460	0.932	0.068

We have performed it again with rotation since the division is clearer. We can see that BL, EM, SF, BS load really high on factor 1 (similar with m=2). AFL, LFF load high on factor 2 (similar with 2 factors). FFF loads negative on all factors (-0.26, -0.70, -0.37). ZST loads the same on both 1 and 2 (0.59, 0.61) and a bit lower on 3 (0.46). The results are very similar to those with m=2 an we can notice how the 3rd factor does not help a lot. As expected the specific variances decreased except for AFL, and the communalities increased. We can visualize the loadings pairwise:



We can notice in these plots how the 1st factor is the more relevant and the 3rd is pretty irrelevant.

For what concern the residual matrix, we expect a better approximation, since the number of factor increased:

	BL	EM	SF	BS	AFL	LFF	FFF	ZST
BL	0.000	-0.002	0.000	0.000	0.000	-0.005	-0.005	0.005
EM	-0.002	0.000	0.002	0.000	-0.001	0.001	-0.003	-0.003
SF	0.000	0.002	0.000	0.000	0.001	0.006	0.019	-0.002
BS	0.000	0.000	0.000	0.000	0.000	0.002	-0.003	-0.002
AFL	0.000	-0.001	0.001	0.000	0.000	0.000	0.009	0.002
LFF	-0.005	0.001	0.006	0.002	0.000	0.000	0.008	-0.007
FFF	-0.005	-0.003	0.019	-0.003	0.009	0.008	0.000	-0.035
ZST	0.005	-0.003	-0.002	-0.002	0.002	-0.007	-0.035	0.000

Showing the residual error we can notice that it is lower than the one for two factors:

Sum squared of residual
0.0039304

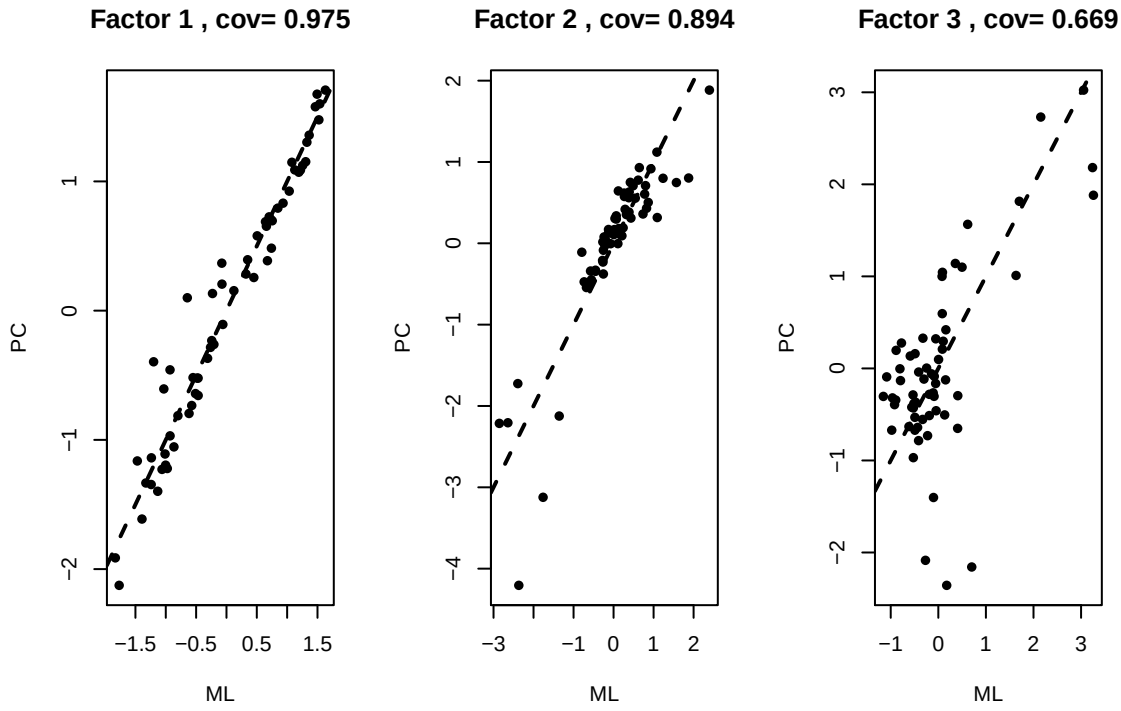
However, we said that these results were expected. The presence of a third factor does a better approximation of the correlation matrix, but has not great loadings, which means that it does not recognize any new group. Furthermore, the increasing proportion of variance explained is not so important:

We can see how the third factor adds a lot less information respect to the first two (0.08 against 0.48 and 0.37 of the first two factors).
From these result we can already conclude that three factors are not useful, we can perform a satisfactory analysis using just two factors, but to be sure of this we can make a comparison with the principal component method.
We start analyzing the eigenvalues, here are the eigenvalues and their cumulative sum (weighted):

	Proportion of variance	Cumulative proportion
Factor1	0.4815033	0.4815033
Factor2	0.3685944	0.8500977
Factor3	0.0765412	0.9266388

Eigenvalues	Cumulative sum
6.3939442	0.7992430
0.9176631	0.9139509
0.4096977	0.9651631
0.1394933	0.9825998
0.0781182	0.9923646
0.0473576	0.9982843
0.0086587	0.9993666
0.0050672	1.0000000

We can see how the first two eigenvalues suggest a factor solution with 2 factors, where they explain 0.91 of the data (0.80 with only one factor is also high but we did not consider it). We also saw, with the proportion of total variance, that the 3rd factor adds less "information" (0.08 against 0.48 and 0.37 of the first two factors). To confirm our hypothesis, we can use also the regression method comparing the PC factor scores with the ML factor scores. If the loading on a particular factor agree, the pairs of scores should cluster tightly about the 45° line through the origin and the correlation be approximately 1. If they do not agree, we can consider to not use that factor.



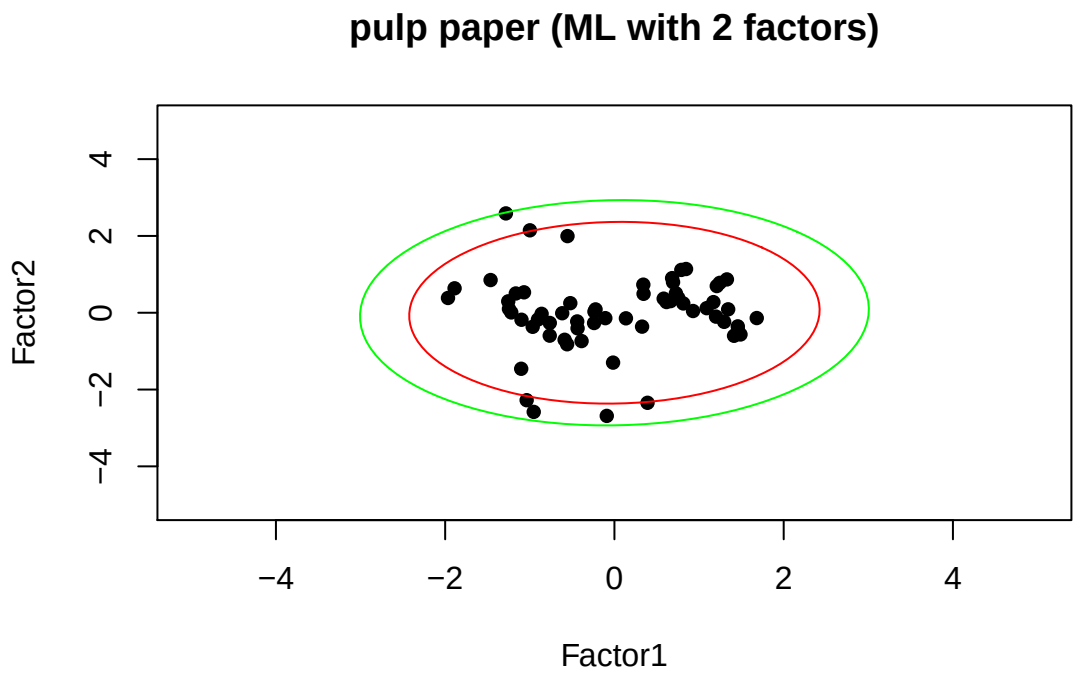
Note that with 3 factors there is still correlation with the ML and PC scores (0.669), but a lot less than the correlation with only 2 factors (0.894). We can conclude that it seems better to use only 2 factors.

Point 1.2

We have seen both the loadings based of none rotation and varimax rotation. Of course varimax rotation is aimed at making clear the difference between the different contribution of the variables to the different factors, so from the loadings rotated we can confirm the separation between "paper properties" (the first four) and "pulp fiber characteristics" (last four). However there are two remarkable facts: the negativity of FFF and the position of ZST. For the first we can reasonably say that in absolute value it belongs to the second factor, so there is no controversial analysis on it. ZST loads high on both factor 1 and factor 2, a little bit more on factor 1, which is controversial since it belongs to the pulp fiber characteristics. We can say that this characteristic, that is proper of the fiber, has some important effect also on the paper directly.

Point 1.3

First of all the factor scores are the estimates of the common factors. In FA we assume that the covariance between the common factors is 0, so we do it also for the estimates. However the regression method is based on the multivariate normality of the variables, and it is usually assumed. In our case we can see that the shape is elliptical and the 0.95 (red line)level of confidence for bivariate normality shows some possible outliers but none of them is over level 0.99(green line), so everything seems fine.



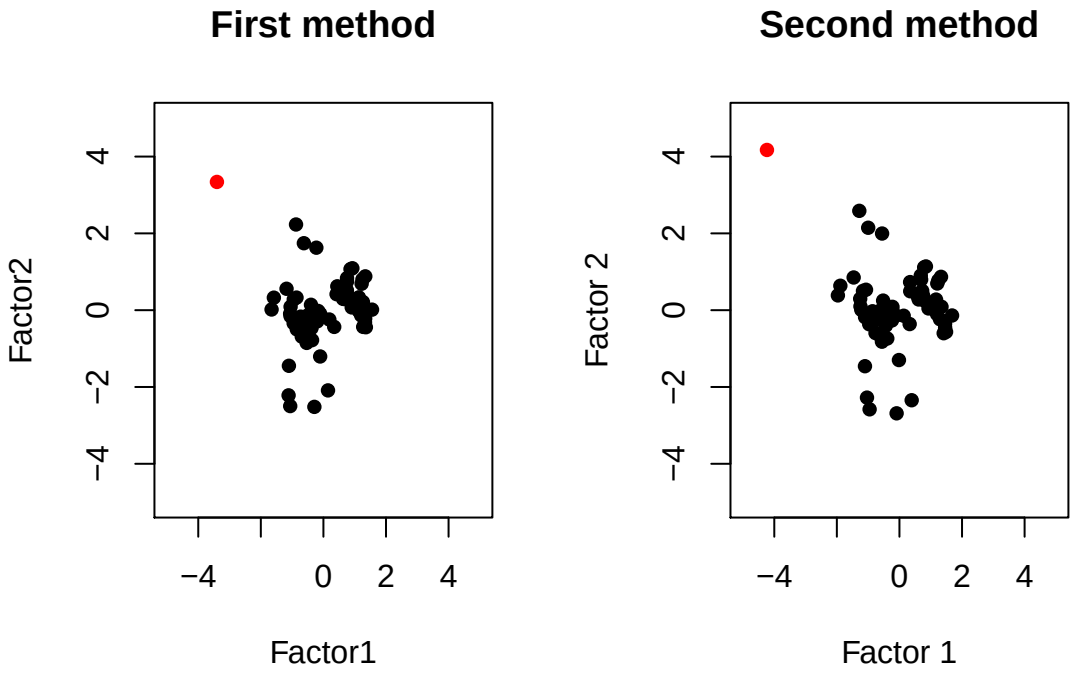
Now we can calculate the correlation matrix and conclude:

	Factor1	Factor2
Factor1	1.0000000	0.0322362
Factor2	0.0322362	1.0000000

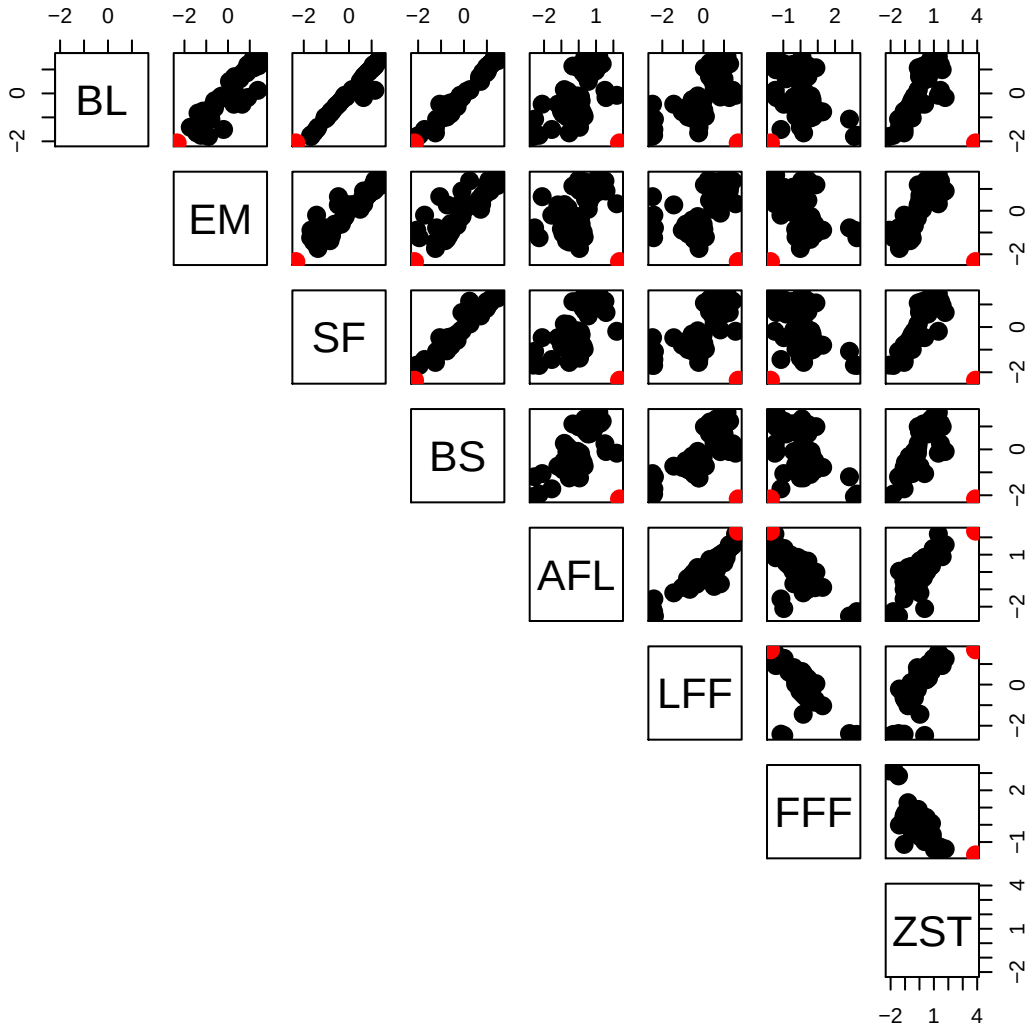
The correlation is little enough to consider that the correlation between the factors is 0, as expected.

Point 1.4

We have to add an observation and calculate its scores. The new observation is : (15.5, 5.5, 2, -0.55, 0.6, 65, -5, 1.2). We could do this in two ways: the first is adding the point directly to the original data and doing again all the analysis. However in this case we would not maintain the previous plot, since mean and standard deviations changed. The second method is calculating directly the score using the regression method, weighting the observation on the already known (original) parameters of pulp paper:



As we can see there are very little changes in the plots, but the conclusions are the same: It is clearly an outlier, we don't have to take a look to the scores value to be convinced of this. We can also see from the paired scatter plot that it seems to be a controversial point from its value: it is probably an outlier not only for the scores (which is clear from the previous plot), but also for the original (scaled) variables.


BL
EM
SF
BS
AFL
LFF
FFF
ZST

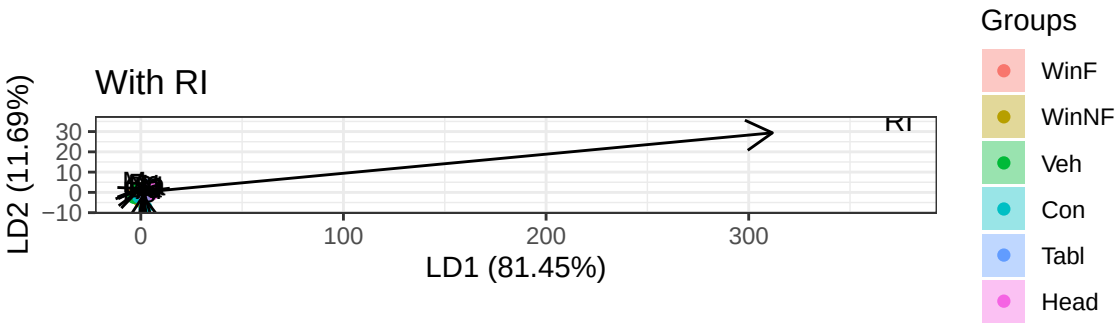
Exercise 2

Point 2.1

We start by performing a linear discriminant analysis.

	LD1	LD2	LD3	LD4	LD5
RI	311.6912516	29.3910394	356.0188308	246.8572080	-804.6553938
Na	2.3812158	3.1650800	0.4596785	6.9243514	2.3987509
Mg	0.7403818	2.9858720	1.5728838	6.8498390	2.8002951
Al	3.3377416	1.7247396	2.2024668	6.4192364	0.9371345
Si	2.4516520	3.0063507	1.7026191	7.5422030	0.9562989
K	1.5714954	1.8620159	1.2861127	8.0761130	2.8209927
Ca	1.0063101	2.3729126	0.6475200	6.6966357	3.7110859
Ba	2.3140953	3.4431987	2.5964981	6.4384927	4.4077058
Fe	-0.5114573	0.2166388	1.2026071	-0.0447494	-1.3029207

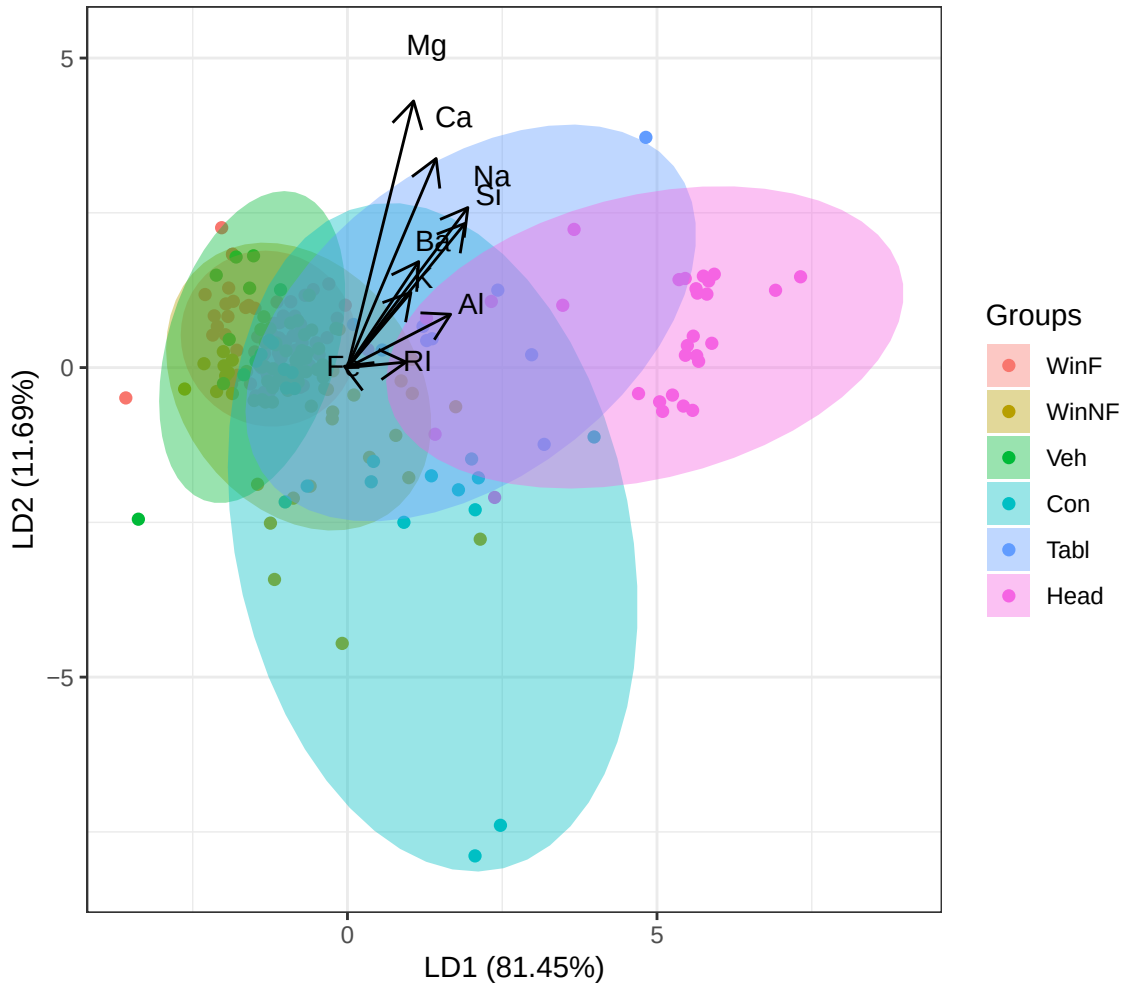
We can see how RI has the biggest weight both in LD1 and LD2. We can confirm it with the biplot (and with a further analysis). Let's see the plot with RI.



If we include RI we obtain a non useful result, as it seems the only one to contribute. It might be true, but in order to see how the linear discriminant coefficients behaves, we have to standardize the data and work with it.

We can see how now Na and Si have the biggest weight in LD1 and Mg and Ca the biggest in LD2. We can visualize it with the biplot:

	LD1	LD2	LD3	LD4	LD5
RI	0.9465639	0.0892566	1.0811807	0.7496717	-2.4436288
Na	1.9445093	2.5846156	0.3753751	5.6544500	1.9588285
Mg	1.0679326	4.3068452	2.2687400	9.8802615	4.0391676
Al	1.6664331	0.8611101	1.0996248	3.2049299	0.4678828
Si	1.8989167	2.3285563	1.3187565	5.8417816	0.7406973
K	1.0249165	1.2143916	0.8387922	5.2671750	1.8398285
Ca	1.4321338	3.3770188	0.9215204	9.5303405	5.2814448
Ba	1.1506127	1.7120247	1.2910288	3.2013426	2.1915962
Fe	-0.0498357	0.0211090	0.1171805	-0.0043603	-0.1269549



Point 2.2

Now let's take a look to the confusion matrix and the relative training error rate

	WinF	WinNF	Veh	Con	Tabl	Head
WinF	52	17	11	0	1	1
WinNF	15	54	6	5	2	2
Veh	3	0	0	0	0	0
Con	0	3	0	7	0	1
Tabl	0	2	0	0	6	0
Head	0	0	0	1	0	25

Training error rate
0.327103

From the confusion matrix we can see that there are so many errors regarding WinF and WinNF, which are also the most "populated" classes in the original data set. Of course these classes can have different features which are shared and represented by the variables, in fact "homogeneous" means that there are less classification error. We have just seen that WinF and WinNF do some error in classifying each other, so we can consider them less homogeneous, but this can be justified by the already cited high number of observations of these classes. For sure the Veh class is not homogeneous, since all its observations are masked. The same for Con, in which almost half of the observations are masked. The most homogeneus seem to be Tabl and Head.

Point 2.3

For the 10-CV we implemented this code, since the partition in groups was already available.

```
# Let's create a list with 10 empty DFs for training 10 models and 10 for validation
# that we will use to make 10 predictions in the 10-fold cross validation.
# We will proceed in this way: we will populate the validation DFs using the partition
# given by groupCV. Then, we will populate 10 DFs for training the model using the data
# not considered by the partition given by groupCV. In this way, each of 10 prediction will
# be made using a partition of a 1/10 of the original glass DF not considered when we will
# train the model.
len <- 10

# validation list
validation = NULL
validation <- vector(mode = "list", length = len)

# training list
training = NULL
training <- vector(mode = "list", length = len)

# Let's populate the validation and training list with 10 empty DFs.
# Same variables as in glass DF.
for (i in 1:len) {
        validation[[i]] = data.frame(matrix(ncol = 10, nrow = 0))
        colnames(validation[[i]]) = c("Ri","Na","Mg","Al","Si","K","Ca","Ba","Fe","type")

        training[[i]] = data.frame(matrix(ncol = 10, nrow = 0))
        colnames(training[[i]]) = c("Ri","Na","Mg","Al","Si","K","Ca","Ba","Fe","type")
}

# Let's populate the DFs considering the assignment determined by groupCV.
```

```
# The DFs for validation are populated using directly the assignment in groupCV,
# the DFs for training the models are populated subtracting each validation DF from
# the original glass DF.

# Assigning each element from glass DF using the assignment from groupCV for the validation DFs
for (i in 1:length(groupCV)) {
        validation[[groupCV[i]]] = rbind(validation[[groupCV[i]]],glass[i,])
}


# Subtracting the validation DFs from glass for the training DFs
for (i in 1:len) {
        training[[i]] = anti_join(glass, validation[[i]])
}

# 10 models, each one for each training DF
models = NULL
models = vector(mode = "list", length = len)

# 10 predictions, each one for each model with its respective validation DF
predictions = NULL
predictions = vector(mode = "list", length = len)

# error matrices for each prediction
err_matrix = NULL
err_matrix = vector(mode = "list", length = len)

# error rates for each error matrices
err_rate = NULL
err_rate = vector(length = len)

# Let's start with our cross validation:
for (i in 1:len) {
        models[[i]] = lda(type~., data=training[[i]])
        predictions[[i]] = predict(models[[i]], validation[[i]])

        # produce the error matrices and each error rate for every matrix
        err_matrix[[i]] = as.matrix(table(predictions[[i]]$class, validation[[i]]$type))
        err_rate[i] = 1-sum(diag(err_matrix[[i]]))/sum(err_matrix[[i]])

        # Output of the 10 confusion matrices with their respective error rates
        #print(list(confusion_matrix = err_matrix[[i]], LDA_error_rate = err_rate[[i]]))
}
```

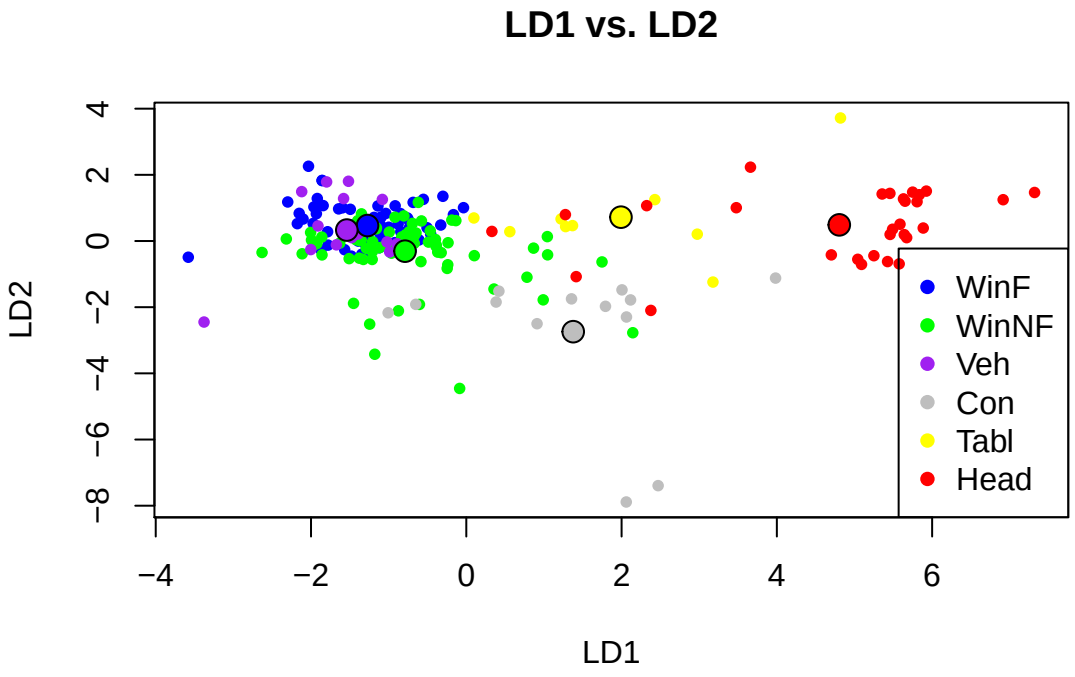
Now that we have our 10 confusion error matrices (each one for each fold) we can sum the elements by their position (the position i,j in the final confusion matrix is the sum of every element in the same position i,j in the previous 10 matrices). Then, we calculate the error of this final error matrix.

	WinF	WinNF	Veh	Con	Tabl	Head
WinF	46	20	11	0	0	1
WinNF	20	49	6	6	3	1
Veh	4	0	0	0	0	0
Con	0	4	0	6	0	2
Tabl	0	2	0	0	5	0
Head	0	1	0	1	1	25

Error of the final confusion matrix
0.38785

The error on the 10 CV matrix is a bit higher than the training error, but this is obvious. What we can do with further analysis is to see if, with this groupCV partition, the error change with the reduced rank LDA. Another analysis we can do is to compare this 10-fold partition given by groupCV and compare it with n random partitions, just to see if this partition is reliable.

Point 2.4



From the analysis done in point 2.2 it seemed that WinF and WinNF were the most populated, so the misclassifications were quite high, but not if we weight on the number of observations. In fact what we can see from this plot is that those two are very near, and so this leads to a misclassification between them, as reflected also in the confusion matrix. Furthermore we can confirm that Veh is completely masked by the two previous cited classes, in fact all the observations are very near WinF and WinNF. One interesting fact is that Table seemed to be the most homogeneus, since there were not many misclassifications, but from this graph it seems completely masked. For Con and Head we can confirm the previous analysis, and even a better one for head, that seems very homogeneus.
To conclude, the less homogeneus are Veh and Table (even if from the confusion matrix we didn't get it),

while, the most homogeneus seems to be Head. A quite good behavior is highlighted for the other classes.

Point 2.5

Now we perform a reduced rank LDA on the original data set, collecting the various error for each dimension:

	Training errors of the reduced rank LDA
R1	0.457944
R2	0.373832
R3	0.364486
R4	0.327103
R5	0.327103

Note that with dimension 5 we had already the result.
Now we perform the same analysis with the 10-CV, so each time for each group we do not perform a full rank lda on the training set, but a reduced rank, and each time we will have a test error for every rank. In this case we have to be aware of the fact that these are test errors, while the previous ones are training errors. This is due to the structure of the 10-CV.

	Error rates with groupCV partition
R1	0.476636
R2	0.411215
R3	0.401869
R4	0.373832
R5	0.387850

We can already see from these tables that the error for the 10 CV is higher, but we have to remember that this method gives us a test error, while the first table gives us training errors. But we will see everything in the following graph, in which we added also another thing: the comparison between the various partitions in 10 groups of our data set.

We implemented an algorithm to produce 100 test errors for every rank, each one with the 10- fold CV method, so that every time, the elements for every fold are randomly sampled from our data set. (not as in groupCV where the partition is already given). We implemented this to see what is the average behavior with the 10-fold CV reduced rank analysis. This is the function we implemented for our analysis:

```
# j is the rank for LDA reduction, K is the number of fold in the CV
cv.lda.rank<- function (dataY, dataX, K=10, j) {

  n <- nrow(dataX)
  i=1
  f <- ceiling(n/K)
  s <- sample(rep(1:K, f), n)
  PvsO=NULL #final confusion matrix

    for (i in 1:K) {
      test.index <- seq_len(n)[(s == i)] #test data
      train.index <- seq_len(n)[(s != i)] #training data

      train.X <- dataX[train.index,]
      test.y <- dataY[test.index,]
      #predicted test set y
      model = lda(type~., data=train.X)
```

```
      lda.pred = predict(model, test.y, dim=j)
      #observed - predicted on test data
      error= mean(lda.pred$class!=test.y$type)
      #error rates
      predvsobs=data.frame(lda.pred$class,test.y$type)
      PvsO=rbind(PvsO,predvsobs)
    }

    resume = list(confusion_matrix = table(PvsO[,1],PvsO[,2]),
                      final_error = 1-sum(diag(table(PvsO[,1],PvsO[,2])))/sum(table(PvsO[,1],PvsO[,2])))

    #Output
    return(resume)
  }
```

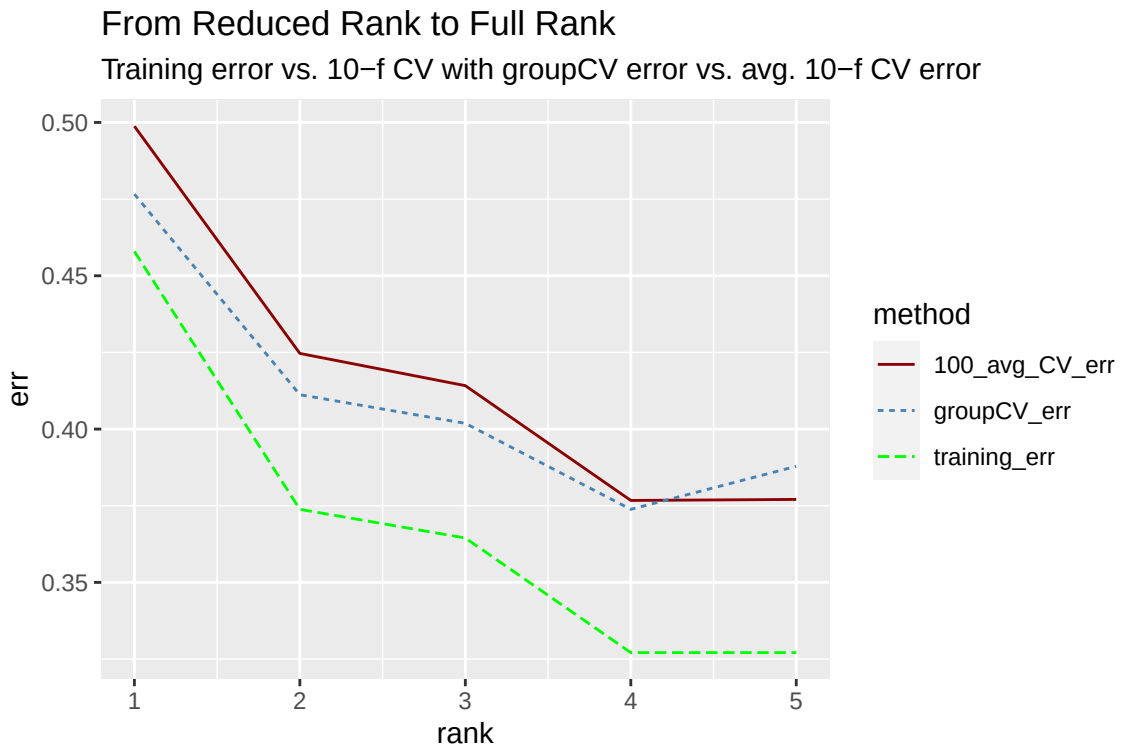
And then we run the simulation:

```
#### LET'S RUN THE 100 SIMULATION
N=100
J=5

row.names <- c(1:N)
rank_err_matrix = array(dim = c(N,J))

for (i in 1:N) {
  #saving the mean error for every rank in the array
  for (j in 1:J) {
    check = cv.lda.rank(dataY=glass, dataX=glass,j=j)
    rank_err_matrix[i,j] = as.numeric(unlist(check[[2]]))
  }
}
```

Let's see the avg for every rank;

	Average error for every rank in our simulation
R1	0.498785
R2	0.424673
R3	0.414159
R4	0.376729
R5	0.377056

Graphically we can see the results it in the following plot.



First of all we can notice that the partition given by groupCV behaves better than the average till rank 4. To conclude we have performed a reduced rank lda on the data and got a training error. The training error is not so interesting, since we know that it's the model created with the training set applied to the training set itself. This is, of course, the best thing we can have, in fact it's the lowest error. However it is more interesting having a test error, which we can have from the 10 CV, performed in reduced rank. This is more interesting because we test our model on observations that were not involved in the training, but gave us higher error (this was predictable). It's important to consider both these errors, to see if in testing there is much more difference than in training, and with this graph we highlighted this difference. We did a further comparison getting nearer a real case study in which we performed our 10 CV on random groups. This lead us to see that groupCV has a lower error (except for full rank) than other groupings. In general going towards the rank returned us decreasing errors. However for training we saw that a full rank and a 4-rank solution perform the same result, while for testing the groupCV full rank error was higher. This makes us prefer a 4-rank solution for our analysis.

To conclude our report we can also answer to some questions: which K is the best for our K-fold partition (with rank 4 for our LDA)? How does the partition given by groupCV behaves with respect to the other K-fold reduced rank 4 LDA analysis? In summary, does groupCV provide a good partition? Let's answer to these questions.

We implemented an algorithm to see which K is the best for our K-fold CV. Every time, the elements for every fold are randomly sampled from our data set (Remember that we choose rank equals to 4 for our LDA):

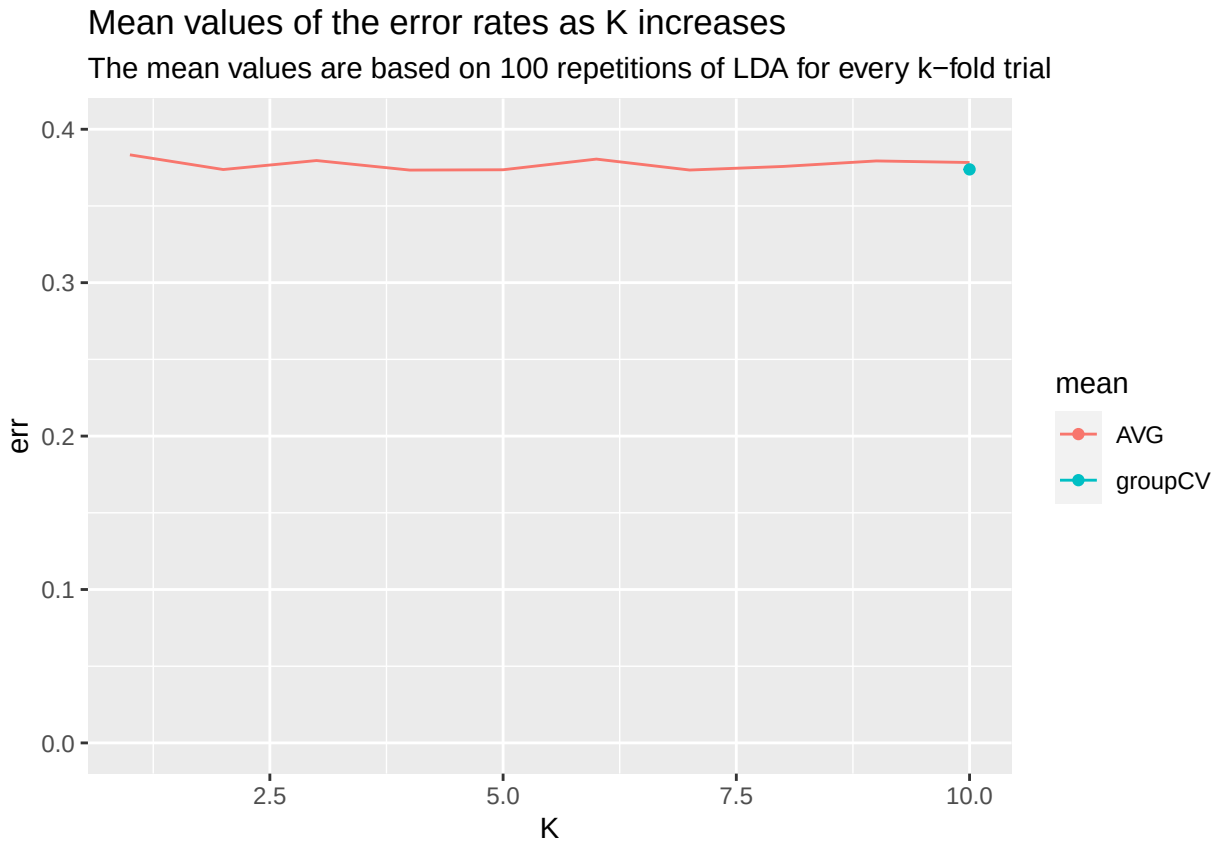
```
cv.lda<- function (dataY, dataX, K=10, seed=123) {

  n <- nrow(dataX)
  #set.seed(seed)
  resume = NULL
  resume = vector(mode = "list", length = K)
  k=1; i=1;

  for (k in 1:K) {

    f <- ceiling(n/K)
    s <- sample(rep(1:K, f), n)
    CV=NULL; PvsO=NULL;

    for (i in 1:k) {
      test.index <- seq_len(n)[(s == i)] #test data
      train.index <- seq_len(n)[(s != i)] #training data

      train.X <- dataX[train.index,]
      test.y <- dataY[test.index,]
      #predicted test set y
      model = lda(type~., data=train.X)
      lda.pred = predict(model, test.y, dimen=4)
      #observed - predicted on test data
      error= mean(lda.pred$class!=test.y$type)
      #error rates
      #CV=c(CV,mean(error))
      predvsobs=data.frame(lda.pred$class,test.y$type)
      PvsO=rbind(PvsO,predvsobs)
    }

    resume[[k]] = list(k_ = k, confusion_matrix=table(PvsO[,1],PvsO[,2]),
                       final_error = 1-sum(diag(table(PvsO[,1],PvsO[,2])))/sum(table(PvsO[,1],PvsO[,2])))
  }

  #Output
  #resume
  return(resume)
}
```

And, also in this case, we ran the simulation 100 times.

We can see how there is not a remarkable difference in choosing a different number K of folds in our K-fold CV.

Let's plot the average error rates for every K with the error rate obtained using the groupCV partition (in which K=10).

	Average error rate for every K in the K-fold CV
K1	0.383330
K2	0.373754
K3	0.379604
K4	0.373376
K5	0.373592
K6	0.380550
K7	0.373417
K8	0.375753
K9	0.379336
K10	0.378318



From the plot we can notice how choosing a number K (from 1 to 10) for the cross validation it is not so relevant. We can also notice that the partition provided by groupCV is slightly better than the average behavior with every K, so we can say it is a "reliable" partition for our analysis.