20092603

STAT0030: ICA3

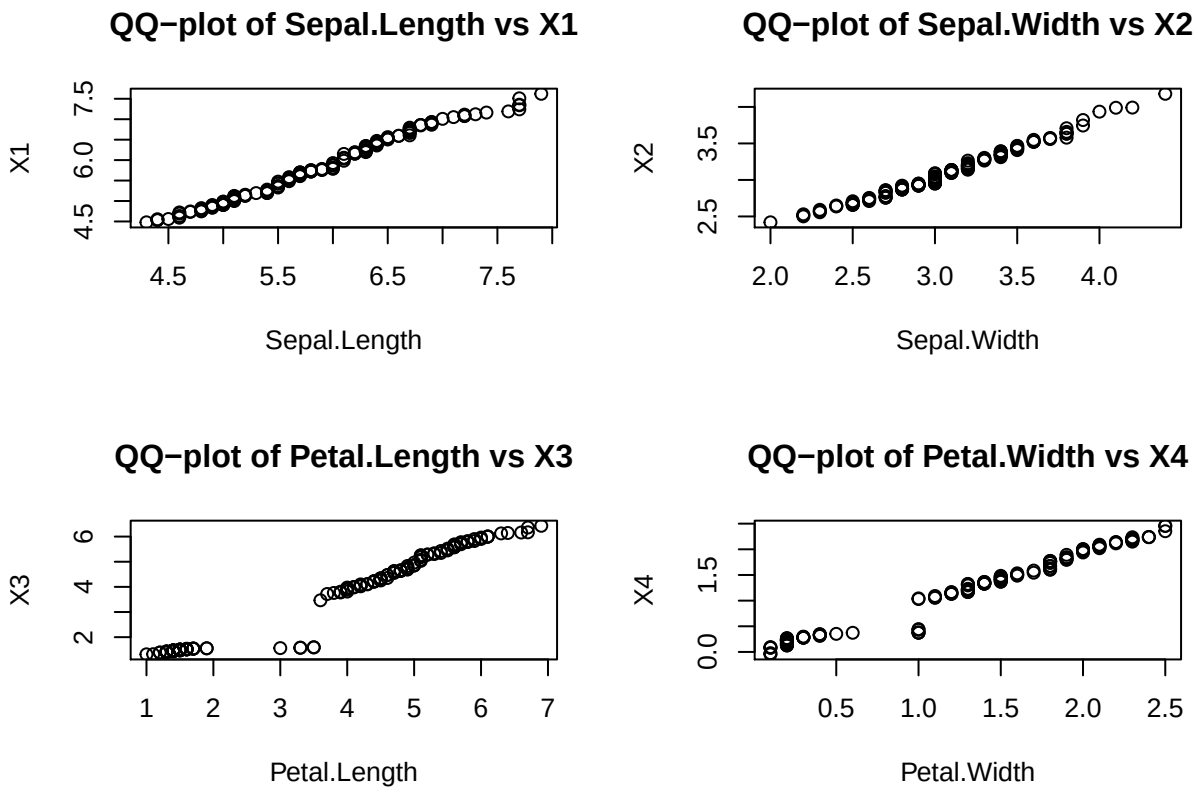
Question 1

```r
set.seed(12345)

# Loading library EMMIXmfa to fit the Mixture of Factor Analyzers model
library(EMMIXmfa)
```

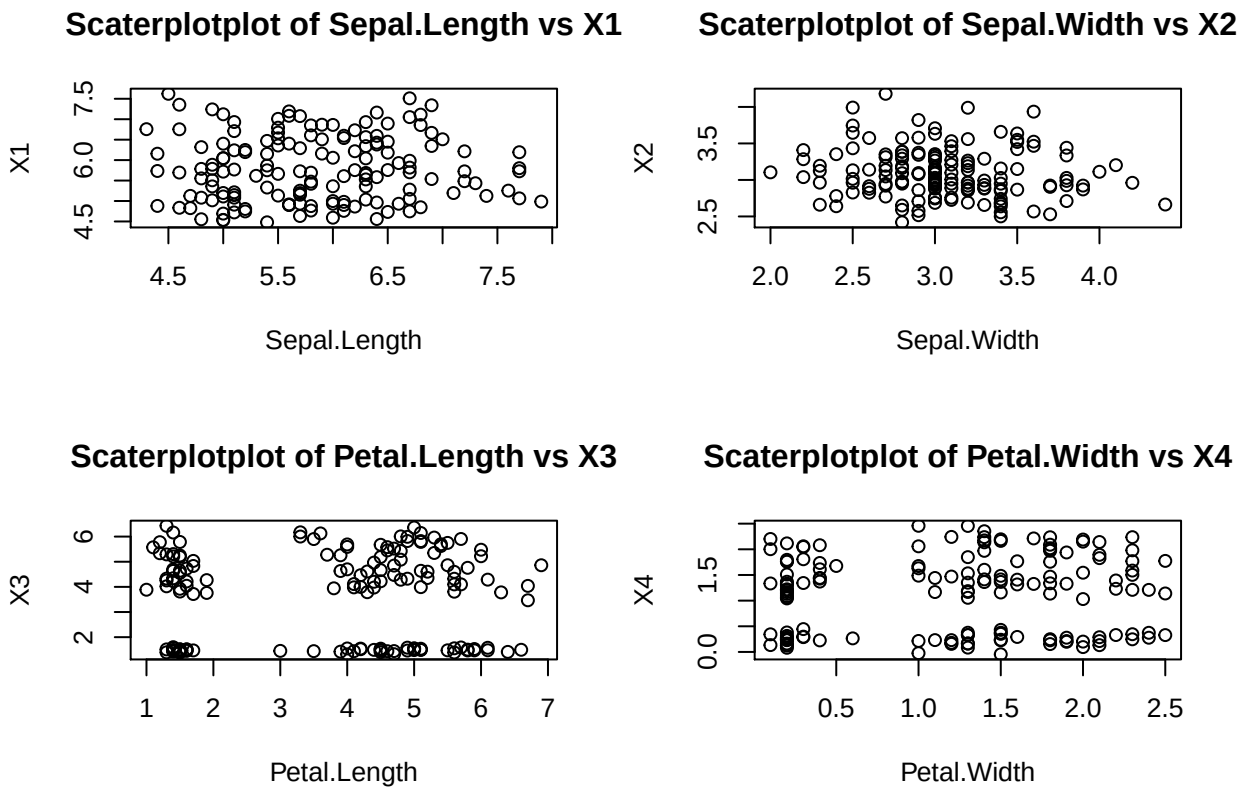
```
## Warning: package 'EMMIXmfa' was built under R version 4.0.5
```

```r
# Fitting a Mixture of Factor Analyzers with M=2 independent standard Gaussian variables
# K=3 Z_i mixture components
# sigma_type has been set to unique
# sigma_type=unique allows the Loading matrices B to be different among different components

dat <- iris[,1:4]   #keeping the first 4 columns of the iris dataset in the variable dat


# fitting a mfa model with K=3 components
# and M=2 factors
# nkmeans=3 and nrandom=0
mfa_model <- mfa(dat , g=3, q=2, nkmeans=3, nrandom = 0,  itmax=500,
      sigma_type = "unique", D_type = "common")

# Function simulate_mfa with arguments N: number of simulated samples and
# mfa_model: output of mfa funciton
# Output: X: matrix NxJ, where J: number of columns in the original dataset

simulate_mfa <- function(N, mfa_model){

  M <- mfa_model$q # number of factors
  vec_M <- seq(1,M) #vector with all values of M

  pi <- mfa_model$pivec #the probabilities of the Zi distribution
  mu <- mfa_model$mu #the K-vectors of J-dimension with the means, # K: number of components
  B <- mfa_model$B #loading matrices, different for each mixture component
  D <- mfa_model$D #Covariance matrix

  J <- nrow(mu) # number of columns of the data
  Z <- seq(1,mfa_model$g) #vector with values of Zi allocation indicator
  #X is the matrix were the simulated observations will be stored
  X <- matrix(nrow = N, ncol = J)
```

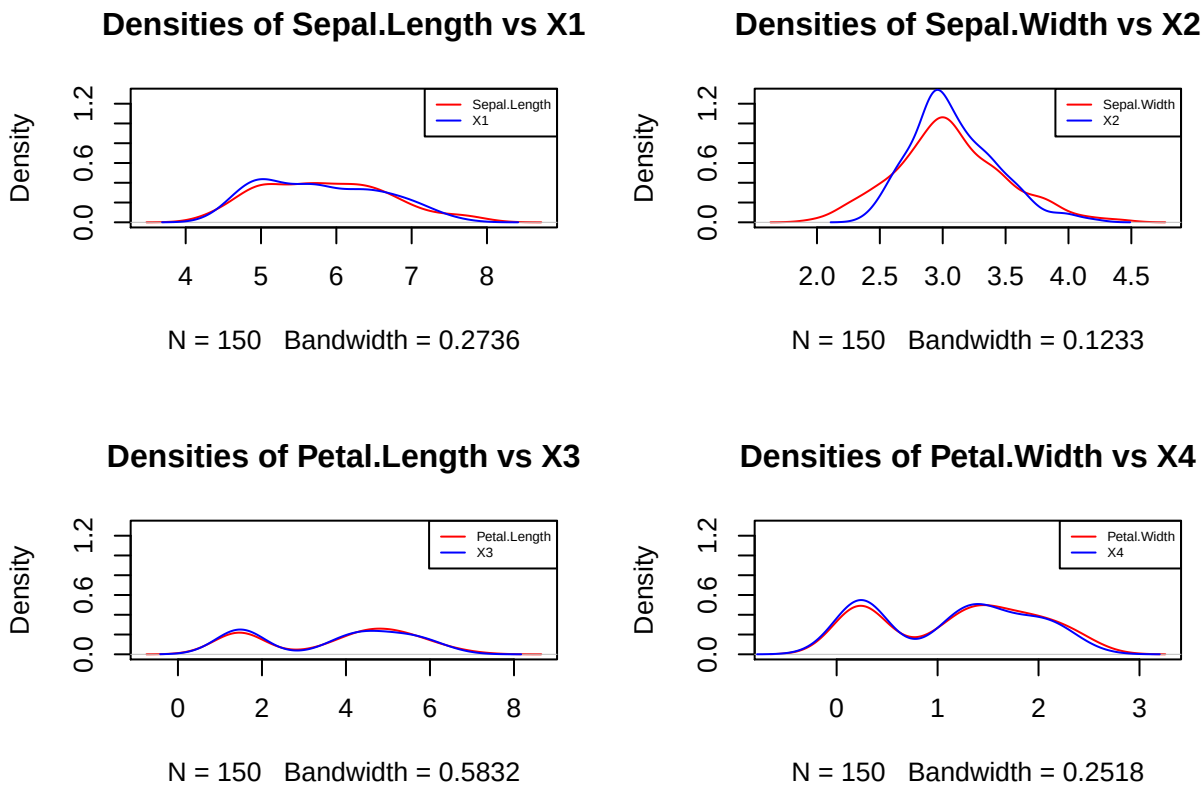
```
  for (i in 1:N){

    Zi <- sample(Z, size=1, prob = pi) # sampling from Zi values with pi probabilities
    Yi <- rnorm(2, mean=0, sd=1) # random sample from standard Gaussian

    for (j in 1:J) {
      #Simulating data points according to the provided formula
      X[i,j] <- mu[j,Zi] + sum(B[j,vec_M,Zi]*Yi[vec_M] ) + D[j,j]

    }

  }

  X
}


#simulating sample equal in size to iris
X <- simulate_mfa(nrow(dat), mfa_model)


# QQ-plots of each variable against the simulated equivalent
# assessing whether they arise from same/similar distributions

par(mfrow=c(2,2))
for (k in 1:4){
  qqplot(dat[,k], X[,k], xlab = names(dat)[k], ylab = paste("X",k,sep="" ),
                main=paste("QQ-plot of",names(dat)[k],"vs",paste("X",k, sep=""), sep=" " ) )
  }
```




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


# Scatterplots of each variable against the simulated equivalent
# assessing how the variables correlate

par(mfrow=c(2,2))
for (k in 1:4){
  plot(dat[,k], X[,k], xlab = names(dat)[k], ylab = paste("X",k,sep="" ),
                main=paste("Scaterplotplot of",names(dat)[k],"vs",paste("X",k, sep=""), sep=" " ) )
  }
```




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

# Plotting the densities of each variable of iris with its simulated variable from simulate_mfa
#  to visually assess the similarity of distributions and hence the quality of the simulation
par(mfrow=c(2,2))
for (k in 1:4){
  plot(density(dat[,k]),type="l",col="red", ylim=c(0,1.3),
     main=paste("Densities of",names(dat)[k],"vs",paste("X",k, sep=""), sep=" " ) )


  points(density(X[,k]),type="l",col="blue")
  legend("topright", legend=c(names(dat)[k], paste("X",k, sep="")),
       col=c("red", "blue"),lty=c(1,1), cex=0.5)
}
```




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

```
##   |                                                                      |
```

Part (b)

In order to visually assess the quality of the simulation we first plot the the quantiles of each variable against the simulated variable, for all four variables of the iris dataset. If the data arise from the same distribution we would expect the scatterplot of their quantiles to form a straight line. For the Sepal Length and Sepal Width variables the QQ-plot is roughly a straight line which suggests that the distribution of the simulated data is similar to the original data. For the Petal width and Petal Length the QQ-plot deviates from being a straight line for what looks like a small cluster of values between 3 and 3.5 for Petal Length and for the value 1 for Petal Width.

To further compare the distributions of each variable with its simulated variable, we plot the empirical densities. It looks like the distributions are very similar for all 4 variables, except perhaps for the Sepal length where the simulated variable has to small modes while Sepal length is flat and for Sepal width the left tail of the simulated variable looks lighter than the original variable. All in all, the simulated variables are fairly close to the original variables. However, by looking at the scatterplots of each variable against the simulated one we can see that for example for the Petal.Length there are a lot of zeros simulated for various values of the variable.

Part (c)

Perhaps, the best model selection method to determine the values of the parameters M and K would be to perform k-fold Cross-Validation. That is to split the data in k sets, use k-1 of them to fit the models for some M and K values and then use BIC as a metric of goodness of fit (or perhaps the Log-Likelihood if the number of parameters in the models is the same) on the validation set that was held out. Repeat k times for different validation set each time and average the value of the BIC (or Log-likelihood). Repeating that for different combinations of M and K values will lead to the best fitting model. Other methods that could decrease the computational cost would be methods based on pruning the number or factors or components until pruning further does not yield better results in terms of BIC (or Log0-Likelihood) or growing the model by increasing the values until the point where increasing M and/or K will not result in a better fit. Finally, a Bayesian approach or a Hierarchical model could be pursued. Also, based on (Kaya, Heysem and A. A. Salah (2015)) the IMoFA and AMoFA algorithms can also be used for model selection.

Question 2

```r
set.seed(12345)

#given the previous mfa model and the data we predict cluster assignments for observations
mfa_labels <- predict(mfa_model, dat )

#kmeans on first four columns of iris with 3 clusters
km <- kmeans(dat , 3)
# kmeans cluster labels
kmeans_labels <- km$cluster

# we then compute the Adjusted Rand Index to assess the clustering of the mfa model
# against the actual labels of the iris dataset, the k-means clustering against the
#iris labels and the mfa model's clustering against the k-means

#comparing the mfa model clustering to the iris Species column
ari_mfa <- ari(mfa_labels, iris[,5])

#comparing the kmeans clustering to the iris Species column
ari_kmeans <- ari(kmeans_labels, iris[,5])

# comparing the agreement between the mfa and kmeans clustering
ari_mfa_kmeans <- ari(mfa_labels, kmeans_labels)

cat("The Adjusted Rand Index for mfa model clustering against iris' labels is", ari_mfa,
    ",\n for the k-means clustering against iris' labels ARI is", ari_kmeans,"and the ARI
    for \n the mfa model clustering against the k-means clustering is",ari_mfa_kmeans,"\n")

#-----------------------------------------------------------------------
# PART C

gaussians <- sapply(1:10, function(x) rnorm(150,0,1)) #generating 10 gaussian variables
# data frame with the first 4 cols from iris and 10 gaussians
iris_noisy <- cbind(iris[,1:4], gaussians)

library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.3
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
#random forest model for the iris_noisy data
rf <- randomForest(x=iris_noisy, y= iris[,5])
#predictors iris + 10 gaussians
#response= iris Species

# the confusion matrix of the prediction (based on OOB data)
cat("Confusion matrix of the predictions based on OOB data \n")
rf$confusion

cat("The Random Forest's Out-Of_bag error is", round( rf$err.rate[500,1], digits=3),"\n")

#mfa model for iris_noisy
mfa_noisy <- mfa(iris_noisy , g=3, q=2, nkmeans=3, nrandom = 0,  itmax=500,
      sigma_type = "unique", D_type = "common")

#given the previous mfa model and the data we predict clusters for observations
mfa_labels2 <- predict(mfa_noisy, iris_noisy )

#comparing the mfa model clustering to the iris Species column
cat("The ARI of the mfa model fitted to iris_noisy is", round( ari(mfa_labels2,
                                                                  iris[,5]), digits=3),"\n")
```

```
## The Adjusted Rand Index for mfa model clustering against iris' labels is 0.941045 ,
##  for the k-means clustering against iris' labels ARI is 0.7302383 and the ARI
##     for
##  the mfa model clustering against the k-means clustering is 0.6912753
## Confusion matrix of the predictions based on OOB data
##            setosa versicolor virginica class.error
## setosa         50          0         0        0.00
## versicolor      0         47         3        0.06
## virginica       0          4        46        0.08
## The Random Forest's Out-Of_bag error is 0.047
##   |                                                                              |
## The ARI of the mfa model fitted to iris_noisy is 0.46
```

Part (b)

Since the ARI index is bounded in $[-1, 1]$, we can report its value as a percentage of similarity. For the mfa model clustering against iris' labels the ARI is 94.1%, while the k-means clustering against iris' labels ARI is 73%. This significant difference in the ARI suggests that the mfa model's clustering is superior to the k-means and it is actually very close to the original labels.

K-means algorithm is fast to fit and easy to implement. However, there are several known weaknesses such as the limitations to relocating the centroids, that usually produce imperfect results. One way to improve the accuracy of the algorithm is to use a better initialization technique. Some of the most common initialization techniques are Random Points, Furthest Point heuristic, Sorting heuristic, Density-based, Projection-based and Splitting techniques, Split algorithm, Maxmin algorithm. According to this study [1], the k means error

was reduced from 15% to 6% when using a better initialization technique. Similar, results are obtained by repeating the k-means algorithm, while using both techniques reduces the error from 15% to 1%.

As already mentioned, the k-means algorithm has the major limitation that it usually fails to optimise the centroid locations globally.And that is because of their inability to move between clusters if the distance is too large or if other stable clusters which are located between the cluster and the cluster they should be moved to, are preventing their movement. As a result the k-means result depends heavily on the initialization and sub-optimal initialization can cause the iterations of the algorithm to end up in local minimums (Fränti, Pasi and Sami Sieranoja (2019)).

Part (c)

The fitted Random Forest (RF) model with 500 trees and 3 variables used at each split has an Out-of-Bag error of 4.7%. To put it differently, the RF model's estimate of its accuracy is 95.3%. OOB error rate is a reliable estimate of the validation error. On the other hand, the ARI of the mfa model using the noisy iris dataset drops significantly, as the clustering of that model is just 46%.

When using the ari function on the predicted label of the mfa model fitted on the iris_noisy, the value we obtain is 0.46. Since the ari value or adjusted Rand index is bounded between -1 and 1, with expected value of 0 if the partitions are random and 1 indicating that the clusterings are exactly the same, we can interpret it as an 46% similarity between the predicted labels by the mfa model on the noisy data and the actual labels of the iris dataset. The Rand index can be seen as a measure of the correct classification decisions made by the model then the Adjusted Rand Index can be interpreted in the same way but corrected-for-chance.

The main difference between the two approaches is that in the mfa case the clustering is performed by maximizing a conditional probability, that is using the data, whereas in the RF case the OOB rate is obtained as a prediction from trees that are not trained on the specific data points. To make it clearer, the RF method uses the labels to compute the OOB error while the mfa model does not see the labels during the fitting process.

Question 3

```r
set.seed(12345)

#Function mfa_impute
#Inputs: data and an output of the mfa function
#Output: list(P_Z, E_X) where P_Z NxK matrix with the probabilities of Z=k, k=1,2,..,K
# given the not NA data
# E_x: NxJxK matrix with the expectations of X_ij given the not NA data and a value k of Z=k

mfa_impute <- function(dat, mfa_model){

  M <- mfa_model$q # number of factors
  vec_M <- seq(1,M) #vector with all values of M

  pi <- mfa_model$pivec #the probabilities of the Zi distribution
  mu <- mfa_model$mu #the K-vectors of J-dimension with the means
  B <- mfa_model$B #loading matrices, different for each mixture component
  D <- mfa_model$D #Covariance matrix
  N <- nrow(dat) # assigning the number of rows of dat to N
  J <- nrow(mu) #columns of the data
  Z <- seq(1,mfa_model$g) #the vector with values of Zi allocation indicator
```

```r
K <- mfa_model$g # number of components

P_Z <- matrix(nrow=N, ncol=K) #initializing P_Z for the first output
E_X <- array(dim=c(N,J,K)) #initializing E_X for the second output

#-------------------------------------
# Calculating the Covariance matrix of X_i using COV = B_Zi*B_Zi^T + D

Sigma <- array(dim=c(J,J,K))

for (l in 1:K) {
    Sigma[,,l] <- B[,,l] %*% t(B[,,l]) + D

}
#-------------------------------------

library(mvtnorm)

for (i in 1:N){

   notNA <- !is.na(dat[i,]) #not NA indices
   x_oi <- dat[i,notNA]

   #-------------------------------------------------
   #Computing the sum of the denominator
   s = 0

   for (z in 1:K){
        s = s + pi[z]*dmvnorm(x_oi, mean =mu[notNA,z] , sigma = Sigma[notNA,notNA,z] )
   }
   #-------------------------------------------------

   for (k in 1:K){

        P_Z[i,k] <- (pi[k]*dmvnorm(x_oi, mean = mu[notNA,k] , sigma = Sigma[notNA,notNA,k]))/ s

   }

}


#Second element of the output, E_X, three dimensional array of size N × J × K
#Entry E X[i, j, k] corresponds to E[X_ij | x_ioi, Z_i = k].

for (k in 1:K){
    for (i in 1:N){

        notNA <- !is.na(dat[i,]) #not NA indices
        x_oi <- dat[i,notNA]

        for (j in 1:J){
            E_X[i,j,k] <- mu[j,k]+ Sigma[j,notNA,k] %*% solve(Sigma[notNA,notNA,k]) %*%
              t(x_oi-mu[notNA,k])
```

```
          }
      }
  }


  list(P_Z = P_Z,E_X = E_X)

}



#---------------------- PART B -------------------------
#chosing random sample of 100 indices
ind <- sample(1:150, size=100, replace = F)
#separating iris dataset
iris1 <- iris[ind,]
iris2 <- iris[-ind,]

# Fitting mfa model on iris1
mfa_model <- mfa(iris1[,1:4], g=3, q=2, nkmeans=3, nrandom = 0,  itmax=500,
      sigma_type = "unique", D_type = "common")

# NAs in iris2, creating iris3
iris3 <- iris2

for (i in 1:nrow(iris2)){
  iris3[i, sample(1:4,size=1)] <- NA
}

#calling the impute function
iris3_imputed <- mfa_impute(iris3[,1:4], mfa_model)
```

```
## Warning: package 'mvtnorm' was built under R version 4.0.3
```

```
#---------------------- PART C -------------------------

# initialising Z_hat to store the highest P_Z value
Z_hat <- rep(NA,nrow(iris3_imputed$P_Z))

#assigning cluster based on max P_Z or conditional probability of Zi=k given the non NA data points
Z_hat <- sapply(1:nrow(iris3_imputed$P_Z), function(x) Z_hat[x]<-which.max(iris3_imputed$P_Z[x,]) )

#predicting class based on mfa model
Z_hat2 <- predict(mfa_model, iris2[,1:4])

# adjusted rand index for the clustering from the imputed method
Z_hat_ari <- ari(Z_hat, iris2$Species)

cat("ARI of the clustering of iris2 with missing values when the highest P_Z value is
selected and for k=1,2,3 and the observation is assigned to the k class:", Z_hat_ari,"\n" )

#adjusted rand index for the clustering from the mfa model on iris1
Z_hat2_ari <- ari(Z_hat2, iris2$Species)
```

```r
cat("ARI of the clustering of iris2 using predict and the mfa model fitted on iris1:", Z_hat2_ari )

#---------------------- PART D -------------------------
#function to compute number of NA values on each row
NA_row <- function(dat){
  f <- matrix(nrow = nrow(dat), ncol = 1)

  for(i in 1:nrow(dat)) {
    f[i] <- sum(is.na(dat[i,]))
  }
  f
}

#initialising EX vector with the conditional expectations of X_ij given the non-NA data
EX <- matrix(nrow=nrow(iris3), ncol = max(NA_row(iris3)  ))

#computing EX
#Because fo the fact that we have just 1 NA value per row EX is computed as a vector
# if EX was NxM with N,M >1 then we can easily modify the code from EX[i] to EX[i,j]

for(i in 1:nrow(iris3)) {
  for (j in 1:ncol(iris3[1:4]) ){

    if ( is.na(iris3[i,j]) ) {
      EX[i] <- sum(iris3_imputed$P_Z[i,1:3] * iris3_imputed$E_X[i,j,1:3])
    }
  }
}


# Computing MSE of the imputed values
mse <- rep(NA, 4 )

for (j in 1:4){

    ind2 <- is.na(iris3[,j])
    mse[j] <- mean( ( iris2[ind2,j]- EX[ind2]  )^2 )

}


# Comparing it to MSE of imputing missing values with empirical mean
mse_empirical <- rep(NA,4)
# calculating means of every column without the NA values
means <- sapply(1:4, function (x) mean(iris3[,x], na.rm = T) )

for (j in 1:4){
    # indices of NAs in j-column
    ind2 <- is.na(iris3[,j])
    #mse of that column if we impute the mean of the column -> NA
    mse_empirical[j] <- mean( ( iris2[ind2,j]- means[j]  )^2 )

}
```

```r
MSEs <- rbind(mse,mse_empirical)
colnames(MSEs) <- colnames(iris3[,1:4])
rownames(MSEs) <- c("MSE of EX imputation", "MSE of mean imputation")

knitr::kable(MSEs)
```

```
##   |                                                                              |
## ARI of the clustering of iris2 with missing values when the highest P_Z value is
## selected and for k=1,2,3 and the observation is assigned to the k class: 0.7701612
## ARI of the clustering of iris2 using predict and the mfa model fitted on iris1: 0.7686454
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
MSE of EX imputation	0.1046013	0.1444966	0.0632878	0.0248183
MSE of mean imputation	0.4815289	0.2169655	1.7764664	0.6831272

Part (c)

The Z_hat clustering i.e.calculating the posterior probabilities of Z_i=k, for k= $1, 2, 3$ and select the highest, on the dataset with missing values, after using the mfa model fitted on 100 iris observations to help us compute the P_Z. We proceed to assign the k index of the highest probability as the class of that observation. The ARI of this approach is 77%. On the other hand, by using the predict funciton and the previous mfa model, fitted on iris1, to perform clustering on the observations of iris2, we get an ARI of 76.86%. These two ARI values are close and we can claim that the clustering performance of the two approaches is similar in terms of accuracy.

Part (d)

When imputing the missing values of the iris3 with the conditional Expectation given the observed data through the mfa model and the function mfa_impute we obtain results that seem very close to zero in terms of MSE. In two of the columns we get MSE values of order $10^(-2)$ and $10^(-1)$ in the other two. In the case of the mean imputation, the MSE of one of the columns is more than 25 times larger and for the rest of the columns, the MSE values are 1.5 to 34 times the values of the first method. We can conclude that so far the mfa model method of imputation produces superior results to the rather naive method of mean imputation.

References

Kaya, Heysem and A. A. Salah. "Adaptive Mixtures of Factor Analyzers." ArXiv abs/1507.02801 (2015): n. pag.

Fränti, Pasi and Sami Sieranoja. "How much can k-means be improved by using better initialization and repeats?" Pattern Recognit. 93 (2019): 95-112.