ICA3-20150879

Contents

Introduction

Data The dataset used is the "famous (Fisher's or Anderson's) iris dataset which gives the measurements in centimetres of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica."

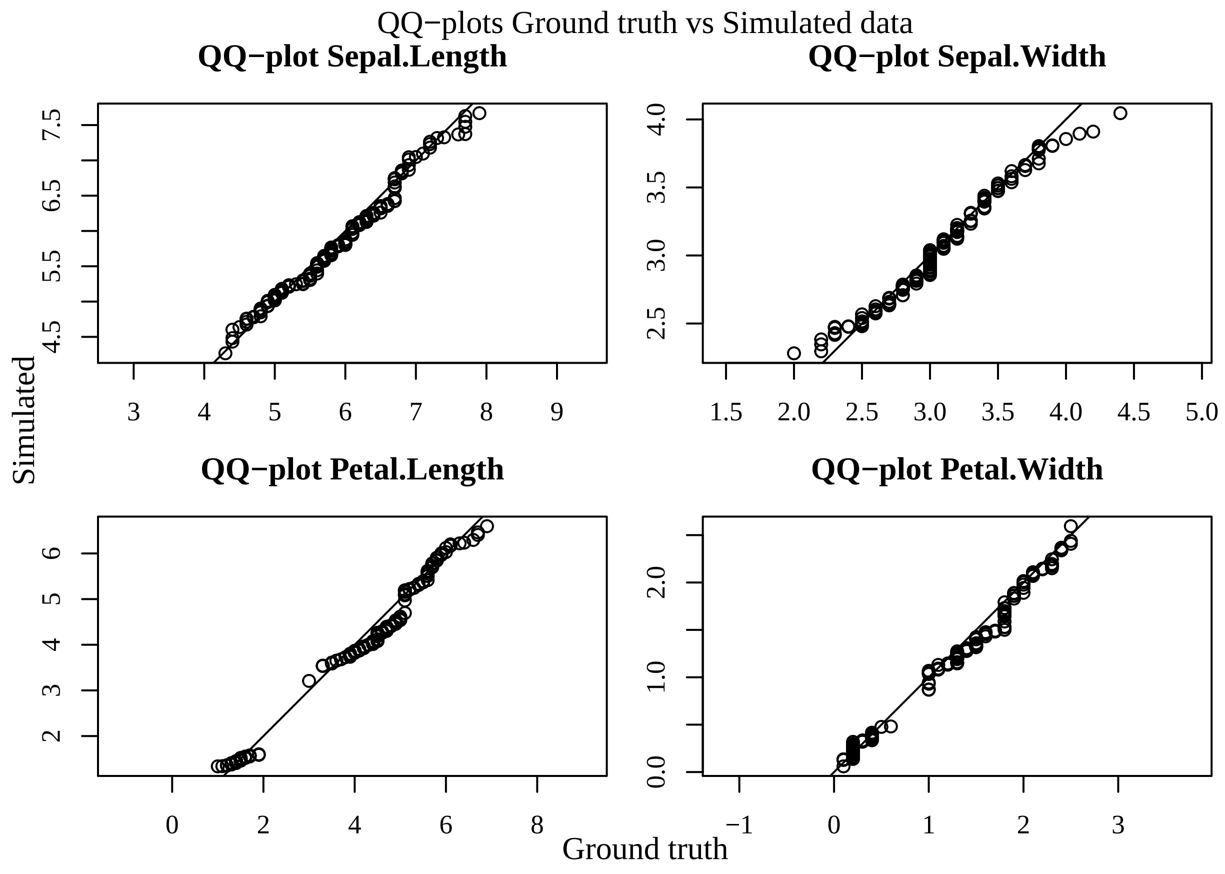
```
#install.packages("EMMIXmfa")
library(knitr)
library(EMMIXmfa)
library(cluster)
library(factoextra)
library(tidyverse)
library(randomForest)
data(iris)
```

Q1. Fitting and evaluating a mfa model

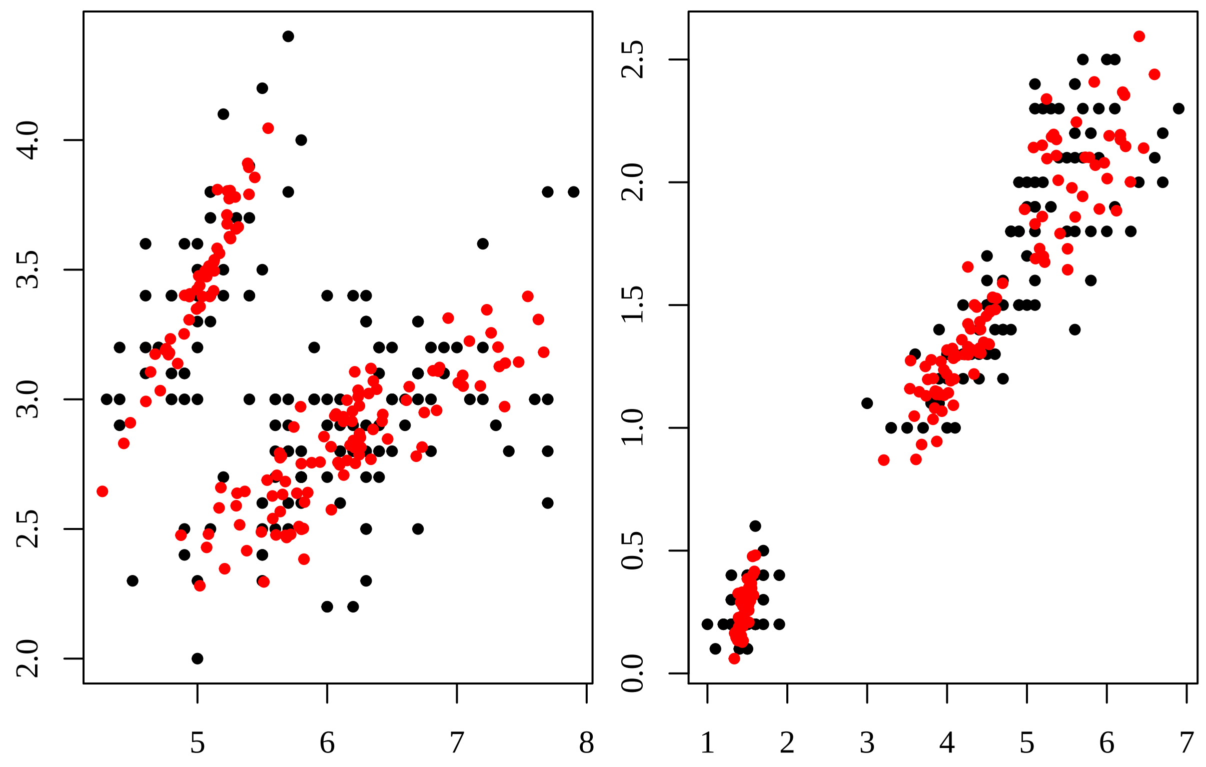
```
set.seed(12345)
#a.1. Fit a mixture of factor analysers with M = 2 and K = 3 for the first four columns
#of iris
M=2 # nubmer of latent factors
K=3 # number of clusters
mfa_mod<-mfa(iris[,1:4],g=K,q=M, sigma_type = "unique",
             D_type = "common",nkmeans = 3, nrandom = 0)
```

```r
#a.2. extract all the corresponding parameters of the fitted model, to simulate data based
#on the model. Program simulation of Y and epsilon

#' @param N, the number of samples
#' @param mfa_model, the output of a call to mfa.
#' @return N x J matrix of simulated samples from mfa_model.
simulate_mfa <- function(N, mfa_model){
  mu<-mfa_model$mu  #Jx1 xK = 4x3
  B<-mfa_model$B    #JxM xK = 4x2x3
  D<-diag(mfa_model$D)    #diag(JxJ x1) = 4X1
  pi_prob<-mfa_model$pivec #1 xK = 1x3

  M<-mfa_model$q # number of latent factors
  K<-mfa_model$g # number of clusters
  J<-nrow(mu) #number of orginal atributes

  z <- sample(1:K,size=N,prob = pi_prob, replace = TRUE)
  X = matrix(nrow = N,ncol =J )
  for (i in (1:N)){
    #print(i)
    zi <- z[i]
    Y <- rnorm(M)
    error <- rnorm(J,sd=D)
    X[i,]<- mu[,zi]+B[,,zi]%*%Y+ error
  }
  return(X)
}

#a.3. Apply simulate_mfa function to the fitted model to simulate a sample of the same
#sample size as iris.
N<-nrow(iris)
mfa.data_sim<-simulate_mfa(N,mfa_mod)

#b. Visually compare the result of this simulation against the real data to provide an
#informal goodness-of-fit assessment.
data_aux<-as.data.frame(mfa.data_sim)
colnames(data_aux)<-colnames(iris[,1:4])
data_aux<-rbind(iris[,1:4],data_aux)
data_aux<-cbind(data_aux,c(rep("real",N),rep("simulated",N)))
colnames(data_aux)[5]<-"data_source"

# b.1. quantile-quantile plots
qq_plots<-list()
par(mfrow=c(2,2), tcl=-0.5, family="serif", omi=c(0.2,0.2,0,0), mai=c(0.3,0.3,0.5,0.2))
for (i in 1:4) {
  col<-colnames(data_aux)[i]
  simulated <- filter(data_aux, data_source %in% c("simulated")) %>%  pull(col)
  real  <- filter(data_aux, data_source %in% c("real")) %>%  pull(col)
  qq_plots[[i]]<-qqplot(x=real, y=simulated, asp=1, main = paste("QQ-plot", col))
  abline( c(0,1))
}
mtext("QQ-plots Ground truth vs Simulated data", side = 3, line = -1, outer = TRUE)
mtext("Ground truth", side=1, outer=T, at=0.5)
mtext("Simulated", side=2, outer=T, at=0.5)


# b.2.scatterplots
par(mfrow=c(1,2))
plot(data_aux[,1:2], col = data_aux$data_source, pch = 20)
plot(data_aux[,c(3,4)], col = data_aux$data_source, pch = 20)
mtext("Real (black) vs Simulated (red) data", side = 3, line = -1, outer = TRUE)
```

```
##    |                                                                              |
```



Real (black) vs Simulated (red) data



Questions Explanation

Q1(b)

In this part we generate new data using factors models. To do so, first a mixture of factor analyser (MFA) was fitted to the 'iris' data. Then using the matrices of the fitted model, we followed the simulation procedure suggested in (. . .)(please see Question pg2).

We can visually analyse the two graphs proposed:

1. QQ-Plots. The plots compare the distributions of both, the real and simulated data. This graph is conventionally used in empirical analysis to compare the actual data distribution to a theorical distribution. In this case, similar distributions are observed, showing only minor differences, in particular departures of the tails of the *Sepal Width.*
2. Scatter Plots. Two scatter plots are presented. Each relates the length and width for the sepal and Petal. Ground true data is represented through the black points, while simulated data is in red.. Graphical evidence show little deviation of the simulated data from the pattern created by the black points.. Therefore, this method is not creating noise.

In conclusion there is no significant difference between the simulated and real data, as the simulated data is contains within the borders of the given ground truth.

Q1(c)

There are several ways to select K, the number of clusters, and M, the number of dimensions, for this model.

Starting with the number of clusters, K. In this case, from an explanatory point of view, it makes sense to use the same number of factors than the variable 'Species' has, so it easy to understand what the model is doin g. Still, it is also possible to define the number of clusters by looking at the data in conjunction with the model considered. While the Elbow Method and/or the Silluete method are applicable, the Bayesian information criterion (BIC) seems more adequate to this case since its performance for classification problem has already been proven. The BIC has also been used in mixture models and the likelihood function for factor models is fairly easy to compute for the factor mixture model (as shown in the questions). The BIC methodology requires to try different numbers of K and compute the BIC for each of the clustering obtained, and finally select the one with the lowest value.

To select the number of components I would suggest to run a Principal Components Analysis (PCA) using the R code 'prcomp' before fitting the model and then see how much each covariable explains the variability of the data using the 'plot' function. This was done during the Lab 8, for this very data (but scaled, as is conventionally implemented for PCA), and it was found that only two variables explain at least 95% of the variance. So simulating data points using M=2 should yield good results. Nevertheless, for prediction purposes we will need to test different Ms for a given K and compute error measurement.

Q2. Use the model for clustering

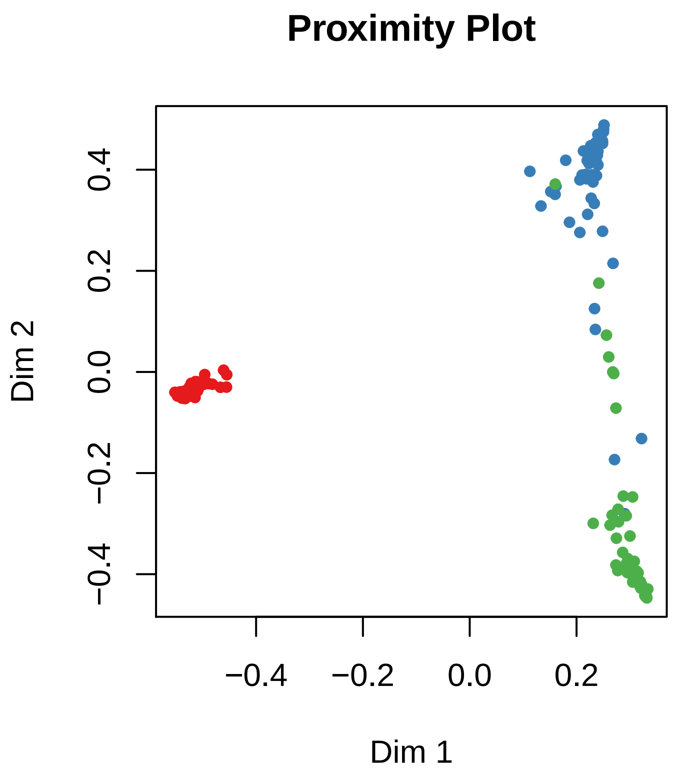
```r
set.seed(12345)
#a.1. Use function predict to generate cluster assignments from the output of mfa as
#applied to the first four columns of iris.
pred_mfa <- predict(mfa_mod, iris[,1:4])

#a.2.Run kmeans on the same, unscaled, columns of iris, again using 3 clusters
kmeanscl <- kmeans(iris[, 1:4], centers = 3, nstart = 10)
pred_kmeans<-kmeanscl$cluster

#a.3. We want to evaluate the agreement between the two clusterings.
#a.3.1. use function ari to compare classifications between the clusters methods and
#against the Species column
cat("\n", "----- Part a)/b) -----",    "\n")
cat("Clasification similarity between MFA and K-Means:",ari(pred_mfa, pred_kmeans),"\n")
cat("Clasification similarity between MFA and Real data:",ari(pred_mfa, iris[, 5]),"\n")
cat("Clasification similarity between MFA and Real data:",ari(pred_kmeans, iris[, 5]),"\n")

#minmis(model$clust, iris[, 5])

#b. Disussion

#c.1. Generate a new dataset iris noisy, made of the first four columns of iris and ten
#columns of independent standard Gaussians.
noise_cols<-10 # number of noise columns
gaussians <- rnorm(nrow(iris)*noise_cols)
noise_matrix<- matrix(gaussians, ncol = noise_cols)
colnames(noise_matrix)<-paste0('noise_', 1:noise_cols)

iris_noisy<-cbind(iris,noise_matrix)
```

```r
#c.2. fit randomForest function to learn and evaluate the ability of this model to
#predict iris$Species from iris noisy
fit_rf <- randomForest(Species ~ ., data = iris_noisy, importance=TRUE, proximity=TRUE)

## does RF recognize that noises are not good predictors varibles?
fit_imp <- importance(fit_rf, type = 1)#generate importance table
df<-as.data.frame(fit_imp[order(fit_imp,decreasing =TRUE),])#order table
colnames(df)<-colnames(fit_imp)#add format to the importance table
kable(round(df,digits = 3))
#varImpPlot(fit_rf)

# c.3. report out-of-bag error evaluation.
### OOB error
cat("\n", "----- Part c) -----",    "\n")
cat("Random Forest Classification error =",
    fit_rf$err.rate[nrow(fit_rf$err.rate),'OOB'],
    "\n")

### is the forecast separable?
prox_points <- MDSplot(fit_rf, iris_noisy$Species, main="Proximity Plot")  #MDS plot
#as.data.frame(fit_rf$confusion) #Confusion matrix for classification

#c.4. Report on the diference between the random forest evaluation and what the ari
#function reports for mfa also applied to this modified dataset.
pred_rf <- fit_rf$predicted

cat("Clasification similarity between MFA and Random Forest:",ari(pred_mfa, pred_rf),"\n")
cat("Clasification similarity between Random Forest and Real data:",
    ari(pred_rf, iris[, 5]),"\n")
```

```
##
##  ----- Part a)/b) -----
## Clasification similarity between MFA and K-Means: 0.6912753
## Clasification similarity between MFA and Real data: 0.941045
## Clasification similarity between MFA and Real data: 0.7302383
```

	MeanDecreaseAccuracy
Petal.Width	30.077
Petal.Length	29.985
Sepal.Length	15.752
Sepal.Width	11.382
noise_7	3.592
noise_2	2.492
noise_8	1.796
noise_3	0.093
noise_4	-0.330
noise_10	-0.483
noise_6	-0.711
noise_5	-0.884
noise_9	-1.151
noise_1	-1.370

```
##
##  ----- Part c) -----
## Random Forest Classification error = 0.05333333
## Clasification similarity between MFA and Random Forest: 0.8681473
## Clasification similarity between Random Forest and Real data: 0.8508387
```



Questions Explanation

Q2(b)

In the console excerpt above, I report the classification similarities obtained using the function 'ari'. We can see that MFA is very similar to the ground truth (~95%) while k-means has greater differences with it (~70%). One way to improve the k-mean model, is to standardise/scale each variable by dividing by its variance. In doing so, the k-means model can better weigh the importance of the different variables used to compute the classification, as it would be less affected by the unit of measurement used.

Q2(c)

The results are plotted above. We can see that the Random Forest model obtained better errors than k-means. Additionally, whilst it only has a ~5% OOB error, the differences with the ground truth are ~85%, which is again better than k-means but worse than MFA. The 'Proximity plot' reveals that, with some exceptions still, the points can be very easily separated with respect to the original class, suggesting that the Random Forest model is good.

The table above shows the role each variable plays in the computation of the OOB error. We can see that the Random Forest model was able to identify that the original variables are more important than the other ten noise gaussian added. Indeed, the original number of variables is above 11, while the noise ones are below 3. Moreover, six variables have a negative impact on the OOB error, suggesting that they are not adding useful information to the model.

Q3. Data Imputation

```r
set.seed(12345)
#a.1. write a function mfa_impute
imp_mfa_conditional_dist <- function(vector, NAs, mu, Cov, K, pi_probab){
  #' Computes the P(Zi = k | xioi) and return a vector of probabilites of the vector
  #belonging to Zi
  x_oi = vector[!NAs]
  mu_oi = mu[!NAs,]
  Cov_oi = Cov[!NAs,!NAs,]
  J_oi<-nrow(Cov_oi)

  prob=matrix(nrow = 1,ncol = K)
  for (k in (1:K)){
    aux <- as.matrix(x_oi-mu_oi[,k])
    aux <- t(aux)%*%solve(Cov_oi[,,k])%*%aux
    aux<- (2*pi)**(-J_oi/2)*det(Cov_oi[,,k])**(-0.5)*exp(-aux/2)
    prob[k]<-pi_probab[k]*aux
  }
  prob<-prob/sum(prob)
  return(prob)
}

imp_mfa_expectation_comp <- function(vector,NAs, mu, Cov,K){
  #Computes the expectation for the missing values.
  x_oi = vector[!NAs]
  mu_oi = mu[!NAs,]
  Cov_oi = Cov[!NAs,!NAs,]
  Cov_j_oi = Cov[,!NAs,]

  J<-nrow(Cov)

  expect<-matrix(nrow = J,ncol = K)
  for (k in (1:K)){
    aux <- as.matrix(x_oi-mu_oi[,k])
    aux<-Cov_j_oi[,,k]%*%solve(Cov_oi[,,k])%*%aux
    expect[,k] <- mu[,k]+aux
  }
  return(expect)
}

#'@param dat a data frame which may have any possible combination of NA entries. N rows
#'@param mfa_model the output of a call to mfa. K # of mixture components
#'@return  list with two entries: a) P_Z, a matrix of size N x K such that P Z[i, k]#
#corresponds to the computed value of P(Zi = k | xioi); b) E_X, and consist of a three-
```

```r
#dimensional array of size N J K, where J is the number of columns in data entry
#E X[i, j, k] corresponds to E[Xij | xioi ;Zi = k].
mfa_impute <- function(dat, mfa_model){
  mu<-mfa_model$mu  #Jx1 xK
  B<-mfa_model$B    #JxM xK
  D<-diag(mfa_model$D)    #diag(Jx1)
  pi_prob<-mfa_model$pivec #1 xK

  N<-nrow(dat)
  M<-mfa_model$q # number of latent factors
  K<-mfa_model$g # number of clusters
  J<-nrow(mu) #number of orginal atributes
  #NAs<-is.na(dat)

  #Compute Covariances matrices
  Cov <- array(NA, dim=c(J,J,K))
  for (i in (1:K)) {
    Cov[,,i]<-B[,,i]%*%t(B[,,i])+diag(D)
  }

  #Initialize outputs
  P_Z <- matrix(nrow = N,ncol = K)
  E_X <- array(NA, dim=c(N,J,K)) #NxJxK

  for (i in (1:N)) {
    vector<-dat[i,]

    NAs<-is.na(vector)
    #
    P<-imp_mfa_conditional_dist( vector, NAs, mu, Cov, K, pi_prob)
    P_Z[i,]<-P

    expec<-imp_mfa_expectation_comp(vector, NAs, mu, Cov, K) #JxK
    E_X[i,,]<-expec
  }
  output <- list()
  output$P_Z <- P_Z
  output$E_X <- E_X
  return(output)
}

#b.1 Randomly partition iris into two datasets, iris1 with 100 data points and iris2 with
#the remaining 50 data points.
set.seed(12345)
n_train<-100
index<-sample((1:nrow(iris)),size=n_train,replace = FALSE)
iris1<-iris[index,]
iris2<-iris[-index,]

#b.2 Fit mfa to the data iris1
M=2 # nubmer of latent factors
K=3 # number of clusters
mfa_model<-mfa(iris[,1:4],g=K,q=M, sigma_type = "unique",
               D_type = "common",nkmeans = 3, nrandom = 0)

#b.3 add NAs to iris2
iris3<-iris2 #intiate iris3
na_index<-sample((1:4),nrow(iris2),replace = TRUE) #randomly select one column per row
for (i in (1:length(na_index))) {
  iris3[i,na_index[i]]<-NA
}
#b.4. run mfa_imputed for iris3
iris3_imputed<-mfa_impute(iris3[,1:4],mfa_model)

#c.
##c.1 create Z_hat, a vector with arg_max P_Z cluster for each row.
Z_hat<-max.col(iris3_imputed$P_Z, 'first')

##c.2 Using predict, generate the vector Z_hat2 of cluster allocations for mfa_model
#and iris2.
Z_hat2 <- predict(mfa_model, iris2[,1:4])

##c.3 Using the ari function, print a comparison of how well each of Z_hat and Z_hat2
#agrees with the classification provided by iris2$Species.
cat("----- Part c) -----",    "\n")
cat("Clasification similarity between MFA and imputed conditional
    distribution:",ari(Z_hat2, Z_hat),"\n")
cat("Clasification similarity between MFA and Real data:",ari(Z_hat2, iris2[, 5]),"\n")
cat("Clasification similarity between MFA and imputed
    conditional distribution:",ari(Z_hat, iris2[, 5]),"\n")

##c.4 Provide a short comment about your findings (recommended max. of 100 words).

#d.
##d.1 compute E[Xij | xioi ] for iris3

N<-nrow(iris3_imputed$E_X)#Dimesion of iris3
J<-ncol(iris3_imputed$E_X)#Dimesion of iris3
E_iris3<-matrix(nrow=N,ncol=J)
for (i in (1:N)) {
  #compute matrix multiplication for each of them
  E_iris3[i,]<-(iris3_imputed$E_X[i,,]%*%iris3_imputed$P_Z[i,])
}

##d.2 for each column j, print the mse between the estimate and the ground truth
#stored in iris2. ignoring entries that observed the column j
MSE_imputed<-array(dim=J)
for (j in (1:J)) {
  na_entries <- na_index==j
  MSE <- iris2[na_entries,j]-E_iris3[na_entries,j]
  MSE <- MSE**2
  MSE_imputed[j] <- mean(MSE)
}
MSE_imputed<-as.data.frame(MSE_imputed, row.names = colnames(iris2[,-5]))
```

```
##d.3 Compare how well you do against estimating each missing Xij by just using
#the empirical average of the observed entries in iris3[, j].
MSE_avg<-array(dim=J)
for (j in (1:J)) {
  na_entries <- na_index==j
  MSE <- iris2[na_entries,j]-mean(iris3[!na_entries,j])
  MSE <- MSE**2
  MSE_avg[j] <- mean(MSE)
}
MSE_imputed$MSE_avg<-MSE_avg
MSE_imputed$MSE_diff<-MSE_imputed$MSE_avg-MSE_imputed$MSE_imputed
colnames(MSE_imputed)<-c("Expectation", "Average", "MSE Diff.")
kable(round(MSE_imputed,digits = 3))
##d.4 Provide a short comment about your findings (recommended maximum of 100 words).
```

```
##   |                                                                              |
## ----- Part c) -----
## Clasification similarity between MFA and imputed conditional
##     distribution: 1
## Clasification similarity between MFA and Real data: 0.936731
## Clasification similarity between MFA and imputed
##     conditional distribution: 0.936731
```

	Expectation	Average	MSE Diff.
Sepal.Length	0.128	0.570	0.442
Sepal.Width	0.111	0.189	0.078
Petal.Length	0.045	3.553	3.507
Petal.Width	0.006	0.669	0.663

Questions Explanation

Q3(c)

We can see that MFA has similarities with the ground truth of ~93%, while the imputed conditional distribution successfully predicted ~77% of the entries. This ~15% difference can be explained by the fact that MFA included all four variables, while the imputation only used three. Additionally, since the similarity between the predictions of the models is ~82%, we can say that there are cases where both models' predictions were wrong.

Q3(d)

The table above summarizes the MSE errors for each variable imputation method, using both, the expectation imputation and replacement by the mean. Overall, we can see that, across all variables, using a mixture of factor analysers to impute missing values does yield higher performance than imputing the missing values with the variable mean. This is particularly relevant in the instance of the the variable Petal Length, where the mean replacement MSE is a hundred times higer than the expectation method.