EM algortihm for mixtures models in R

Sebastian Calcetero

April 9 2021

1. Mixtures models

We say that a random variable X follows a *mixture distribution* if its density function is of the form

$$f(x;\Phi) = \sum_{j=1}^{m} a_j f_j(x;\theta_j)$$

where $f_j(x;\theta_j)$ are densities, called the *components*, and a_j are non-negative coefficients that add up to 1, called the *weights*. Here Φ is the vector of all parameters.

A mixture model has the following important interpretation: Suppose the whole population is divided in m classes, where the j-th class represents the $100 * a_j\%$ of the population and the distribution of the random variable X in such a class is given by $f_j(x;\theta_j)$. Then the density of the random variable X for an individual selected at random from the population is given by the mixture distribution above.

2. EM algorithm for fixture models

Suppose we observe the values of the random variable X_i for a total of n individuals. In addition, consider the unobserved random variables Y_i which denote the class from which the i-th individual was selected, so that $P(Y_i = j) = a_j$.

The *complete* data log-likelihood is then,

$$l(\Phi;\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{1}_{\{Y_i=j\}} \log\left(a_j f_j(x_i;\theta_j)\right)$$

In the EM algorithm the vector Φ is estimated iteratively, for which we use the notation $\Phi^{(k)}$. Hence, given a starting value $\Phi^{(0)}$, the algorithm iterates over these steps:

- **E-Step:** Take the conditional expectation of the complete lok-likelihood with respect to the observed data $\mathbf{X}$ and the previous value of the parameters $\Phi^{(k-1)}$

$$Q(\Phi;\mathbf{X},\Phi^{(k-1)}) = E(l(\Phi;\mathbf{X},\mathbf{Y})|\mathbf{X},\Phi^{(k-1)}) = \sum_{i=1}^{n}\sum_{j=1}^{m} g_{ij}^{(k-1)} \log\left(a_j f_j(x_i;\theta_j)\right)$$

where $g_{ij}^{(k-1)} = E(\mathbf{1}_{\{Y_i=j\}}|\mathbf{X},\Phi^{(k-1)}) = P(Y_i = j|\mathbf{X},\Phi^{(k-1)}) = a_j^{(k-1)} f_j(x_i;\theta_j^{(k-1)})/f(x_i;\Phi^{(k-1)})$ are the posterior probabilities of belonging to a class.

- **M-Step:** We maximize the previews expression to obtain the values of the parameters for the next iteration, $\Phi^{(k)}$. That is, $\Phi^{(k)} = argmax_{\Phi} Q(\Phi;\mathbf{X},\Phi^{(k-1)})$. Taking derivatives this yields to the equations,

$$a_j^{(k)} = \frac{1}{n}\sum_{i=1}^{n} g_{ij}^{(k-1)}$$

$$\sum_{i=1}^{n} g_{ij}^{(k-1)} \frac{\partial}{\partial \theta_j} \log\left(f_j(x_i;\theta_j)\right) = 0, \quad j = 1, \ldots, m$$

The last expression depends in the components selected, and might be solved analytically or numerically.

These steps are repeated until convergence to a solution.

3. Example EM algorithm for mixture of exponential distributions

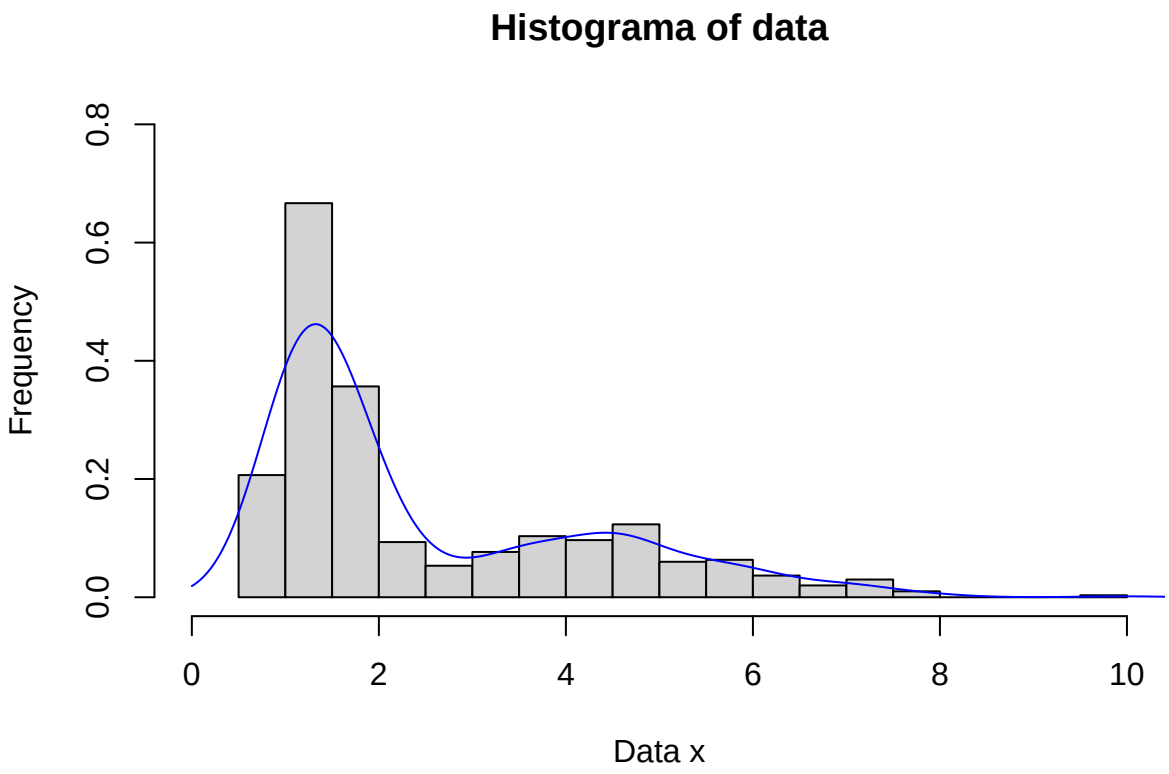
Mixture of exponential distributions

For the exponential distribution we have $f_j(x;\theta_j) = \frac{1}{\theta_j} e^{-x/\theta_j}$, and so

$$f(x;\Phi) = \sum_{j=1}^{m} a_j \frac{1}{\theta_j} e^{-x/\theta_j}$$

Let's generate some data first

```
#Let's get some data ...
n <- 600 #Sample size
set.seed(201111754)
X <- data.frame(x= exp(c(rnorm( n = n/3, mean = 0.1,  sd = 0.2),
                         rnorm( n = n/3, mean = 0.5,  sd = 0.2  ),
                         rnorm( n = n/3, mean = 1.5,  sd = 0.3  ))))

hist(x = X$x,
     main= "Histograma of data",
     xlab = "Data x",
     ylab = "Frequency",
     breaks = 20,
     freq = F,
     ylim = c(0,0.8),
     xlim = c(0,10))
lines(density(X$x,from = 0), col="blue")
```



Starting values for EM

One way to get some starting values for the algorithm is by first generating an artificial clustering of the data in m groups, and then estimate the parameters for each group using traditional MLE techniques.

```
#Find initial values for the EM

m <- 4  #NUmber of components for the mixture

cluster <- kmeans(x = X,
                  centers = m) #Clustering using k means

initial <- X %>% cbind(group = cluster$cluster) %>%
              group_by(group) %>%  #Summarize by group
              summarise( theta = mean(x), #MLE of theta is the sample mean
                         a =  n() / dim(X)[1]) #Estimate a as the sample proportion

#Store values
Phi_0 <- list(Theta = initial$theta,
              a = initial$a )

initial #Display Values
```

```
## # A tibble: 4 x 3
##   group theta     a
##   <int> <dbl> <dbl>
## 1     1  4.16 0.208
```

```
## 2     2  6.21 0.102
## 3     3  1.98 0.212
## 4     4  1.20 0.478
```

E-Step

Therefore the E-step takes the form

$$Q(\Phi; \mathbf{X}, \Phi^{(k-1)}) = \sum_{i=1}^{n} \sum_{j=1}^{m} g_{ij}^{(k-1)} \log \left(a_j \frac{1}{\theta_j} e^{-x/\theta_j} \right)$$

where the posterior probabilities are given by

$$g_{ij}^{(k-1)} = \frac{a_j^{(k-1)} \frac{1}{\theta_j^{(k-1)}} e^{-x/\theta_j^{(k-1)}}}{\sum_{h=1}^{m} a_h^{(k-1)} \frac{1}{\theta_h^{(k-1)}} e^{-x/\theta_h^{(k-1)}}}$$

```
# E - Step: Compute the posterior probabilities

f_joint = function(x,y, Phi){ Phi$a[y]*dexp(x = x,rate = 1/Phi$Theta[y]) } #Joint distribution

g_num <- outer(X = X$x,
               Y = 1:m,
               Phi = Phi_0,
               FUN = f_joint ) #Compute the numerator of g for all data

g_denom <- apply(X = g_num,
                 MARGIN = 1,
                 FUN = sum ) #Compute the denominator of g for all data

g_0 <-g_num/g_denom #g is the ratio

colnames(g_0) = outer("g^(0)_,",1:m,FUN = paste0) #Names
rownames(g_0) = outer("X_",1:n,FUN = paste0)

head(g_0) #Display
```

```
##      g^(0)_,1   g^(0)_,2  g^(0)_,3  g^(0)_,4
## X_1 0.1257581 0.04383518 0.2158349 0.6145718
## X_2 0.1222261 0.04237491 0.2135664 0.6218326
## X_3 0.1331582 0.04692698 0.2203486 0.5995662
## X_4 0.1388876 0.04935052 0.2236305 0.5881314
## X_5 0.1450909 0.05200361 0.2269843 0.5759212
## X_6 0.1320423 0.04645799 0.2196882 0.6018115
```

M-Step

For this particular case, as covered in the lecture, the set of equations in the M-step have a explicit solution:

$$a_j^{(k)} = \frac{1}{n} \sum_{i=1}^{n} g_{ij}^{(k-1)}$$

$$\theta_j^{(k)} = \frac{\sum_{i=1}^{n} g_{ij}^{(k-1)} X_i}{\sum_{i=1}^{n} g_{ij}^{(k-1)}} = \frac{\sum_{i=1}^{n} g_{ij}^{(k-1)} X_i}{n a_j^{(k)}}$$

```
#M-step: Compute updated values for the means and the weights with the formulas

# New weights
a_1 = apply(X = g_0,
            MARGIN = 2, #By columns
            FUN = sum)/n

names(a_1) = outer("a_",1:m,FUN = paste0) #Names

#Weighted average
theta_1 = apply(X = X$x*g_0,
                MARGIN = 2, #By columns
                FUN = sum)/(n*a_1)

names(theta_1 ) = outer("theta_",1:m,FUN = paste0) #Names

Phi_k=list(Theta = theta_1, a = a_1 ) #Store values in alist

Phi_k #Display results
```

```
## $Theta
##  theta_1  theta_2  theta_3  theta_4
## 3.325800 3.650076 2.442058 1.802794
##
## $a
##        a_1        a_2        a_3        a_4
## 0.23178318 0.09989436 0.23218769 0.43613477
```

Iterations of the EM algorithm

To get the estimates we iterate until convergence of the estimation:

```
convergence = FALSE #Boolean that denotes the convergence to a solution
k=1 #Stores the value of the current iteration

while(!convergence){

#E-Step: Posterior probabilities (same code form above)
g_num <- outer(X = X$x,
               Y = 1:m,
               Phi = Phi_k,
               FUN = f_joint )

g_denom <- apply(X = g_num,
                 MARGIN = 1,
                 FUN = sum )

g_k <-g_num/g_denom

colnames(g_k) = outer("g^(k)_",1:m,FUN = paste0) #Names
rownames(g_k) = outer("X_",1:n,FUN = paste0)
```

```r
#M-Step: Compute updated values values in the M-step
a_k = apply(X = g_k,
            MARGIN = 2,
            FUN = sum)/n

theta_k = apply(X = X$x*g_k,
                MARGIN = 2,
                FUN = sum)/(n*a_k)

names(a_k) = outer("a_",1:m,FUN = paste0) #Names
names(theta_k ) = outer("theta_",1:m,FUN = paste0) #Names

#Check convergence
convergence <- (sum((Phi_k$Theta-theta_k)^2)+ #L2 norm between iterations
              sum((Phi_k$a-a_k)^2)<0.00001)

#Print values of iterations
print(paste0("------------------------------------------------------------------ "))
print(paste0("Iteration ", k))
print(paste0("Parameter Theta"))
print(theta_k)
print(paste0("Weights a"))
print(a_k)

#Update new value of parameters
Phi_k=list(Theta = theta_k, a = a_k )
k=k+1

}
```

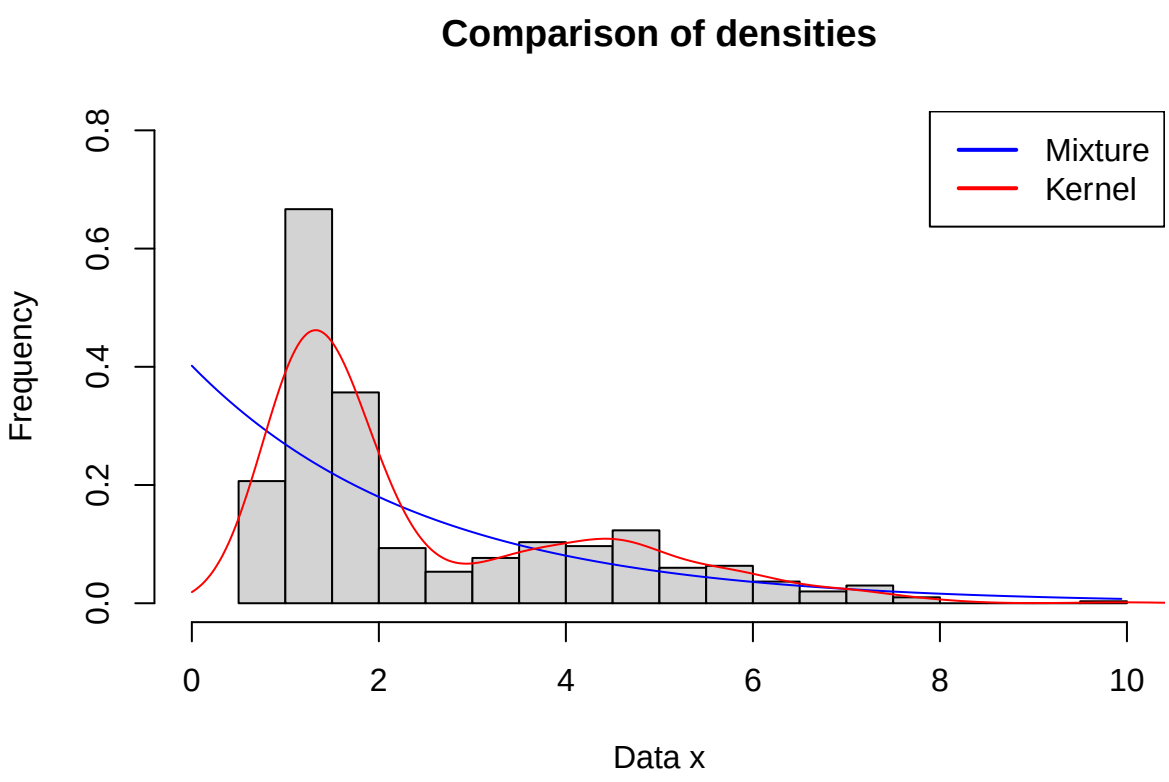
```
## [1] "------------------------------------------------------------------ "
## [1] "Iteration 1"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.886422 2.987132 2.520623 2.141453
## [1] "Weights a"
##        a_1        a_2        a_3        a_4
## 0.23311536 0.09901252 0.23713385 0.43073827
## [1] "------------------------------------------------------------------ "
## [1] "Iteration 2"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.679554 2.719574 2.517174 2.316599
## [1] "Weights a"
##        a_1        a_2        a_3        a_4
## 0.23306980 0.09871984 0.23826139 0.42994897
## [1] "------------------------------------------------------------------ "
## [1] "Iteration 3"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.581081 2.598812 2.505615 2.404109
## [1] "Weights a"
##        a_1        a_2        a_3        a_4
```

```
## 0.23302087 0.09863976 0.23851912 0.42982024
## [1] "------------------------------------------------------------ "
## [1] "Iteration 4"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.533651 2.541917 2.497692 2.447276
## [1] "Weights a"
##       a_1       a_2       a_3       a_4
## 0.23300496 0.09861944 0.23857904 0.42979656
## [1] "------------------------------------------------------------ "
## [1] "Iteration 5"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.510645 2.514593 2.493291 2.468462
## [1] "Weights a"
##       a_1       a_2       a_3       a_4
## 0.23300068 0.09861444 0.23859314 0.42979174
## [1] "------------------------------------------------------------ "
## [1] "Iteration 6"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.499444 2.501352 2.491015 2.478836
## [1] "Weights a"
##       a_1       a_2       a_3       a_4
## 0.23299961 0.09861323 0.23859648 0.42979068
## [1] "------------------------------------------------------------ "
## [1] "Iteration 7"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.493980 2.494907 2.489873 2.483910
## [1] "Weights a"
##       a_1       a_2       a_3       a_4
## 0.23299934 0.09861294 0.23859728 0.42979044
## [1] "------------------------------------------------------------ "
## [1] "Iteration 8"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.491312 2.491764 2.489308 2.486391
## [1] "Weights a"
##       a_1       a_2       a_3       a_4
## 0.23299928 0.09861287 0.23859747 0.42979039
## [1] "------------------------------------------------------------ "
## [1] "Iteration 9"
## [1] "Parameter Theta"
##  theta_1  theta_2  theta_3  theta_4
## 2.490009 2.490229 2.489030 2.487604
## [1] "Weights a"
##       a_1       a_2       a_3       a_4
## 0.23299926 0.09861285 0.23859751 0.42979037
```

Fitted density

Now let's look at the fitted mixture

```r
#Density of the mixture
fitted_density = function(x){ apply(X = outer(X = x,
                                              Y = 1:m,
                                              FUN = f_joint,
                                              Phi=Phi_k),
                                    MARGIN = 1,
                                    FUN = sum)}

#Some values for a grid
xval = seq(0,max(X$x),0.01) #Grid of X values
fval = fitted_density(xval) #Density evaluated

#Plot the fitted densities
hist(x = X$x,   #Plot data
     main= "Comparison of densities",
     xlab = "Data x",
     ylab = "Frequency",
     breaks = 20,
     freq = F,
     ylim = c(0,0.8),
     xlim = c(0,10))
lines(xval,  #Plot fitted density
      fval,
      type="l",
      col="blue")
lines(density(X$x,from = 0),  #Plot fitted kernel
      col="red")
legend('topright',
       col=c("blue", 'red'),
       lwd=2,
       legend=c("Mixture", "Kernel"))
```



#4. Example using Mixtools

We can do this calculation using the `mixtools` package

```
#install.packages("mixtools")
library(mixtools)
```

```
## mixtools package, version 1.2.0, Released 2020-02-05
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

These R package is able to fit several types of mixture models, not only with several distributions but also including more flexible models such as regression type models to account for covariates as well. We'll just do simple fittings here. The name of the function changes depending on what component distributions you want to fit.

Mixture of Exponentials

For the mixture of exponential distributions we can use the function `expRMM_EM`

```
#Fit the model
fit_exp <- expRMM_EM(x = X$x, #Data
                     k = m, #NUmber of components
                     lambda = Phi_0$a, #Starting values for lambda (optional)
                     rate = Phi_0$Theta^(-1) ) #Starting values for the rates (optional)
```

```
## number of iterations = 18
```

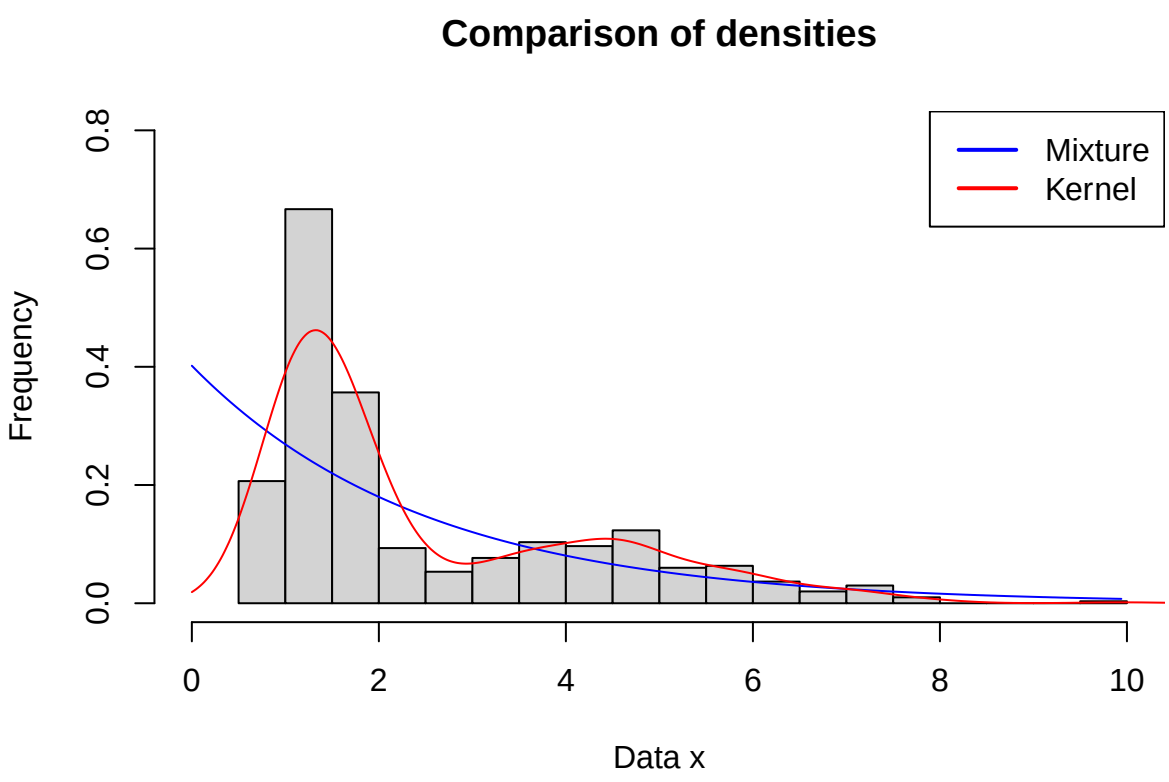
```
#Obtain estimations
lambda <- fit_exp$lambda
```

```r
means <- fit_exp$rate^(-1)

names(lambda) = outer("a_",1:m,FUN = paste0) #Names
names(means) = outer("theta_",1:m,FUN = paste0) #Names

Phi_exp <-list(Theta = means,
                   a = lambda)

#See estimates
Phi_exp
```

```
## $Theta
##  theta_1  theta_2  theta_3  theta_4
## 2.488768 2.488768 2.488764 2.488760
##
## $a
##        a_1        a_2        a_3        a_4
## 0.23299926 0.09861285 0.23859753 0.42979037
```

We can see similar results to what we did by hand in terms of the parameters, and so the same for the fitted distribution:

```r
#Density of the mixture
fitted_density_exp = function(x){ apply(X = outer(X = x,
                                                  Y = 1:m,
                                                  FUN = f_joint,
                                                  Phi=Phi_exp ),
                                        MARGIN = 1,
                                        FUN = sum)}

#Evaluate fitted density
fval_exp=fitted_density_exp(xval)

#Plot the fitted densities
hist(x = X$x,   #Plot data
     main= "Comparison of densities",
     xlab = "Data x",
     ylab = "Frequency",
     breaks = 20,
     freq = F,
     ylim = c(0,0.8),
     xlim = c(0,10))
lines(xval,  #Plot fitted density
      fval_exp,
      type="l",
      col="blue")
lines(density(X$x,from = 0),  #Plot fitted kernel
      col="red")
legend('topright',
       col=c("blue", 'red'),
       lwd=2,
       legend=c("Mixture", "Kernel"))
```



Mixture of Gammas

Let's try to fit a mixture that is more flexible. Let's say for instance a mixture of Gammas. In this case the model we have $f_j(x;\theta_j) = \frac{1}{\Gamma(\alpha_j)\beta_j^{\alpha}} x^{\alpha-1}e^{-x/\beta_j}$, and so

$$f(x;\Phi) = \sum_{j=1}^{m} a_j \frac{1}{\Gamma(\alpha_j)\beta_j^{\alpha}} x^{\alpha-1}e^{-x/\beta_j}$$

To estimate this model using `mixtools` we can use the function `gammamixEM`. Its usage is analogous to the previous one

```
#Fit the model
fit_gamma<- gammamixEM(x = X$x, #Data
                       k = m , #Number of components
                       fix.alpha = T, #Same shape parameter (optional)
                       lambda = Phi_0$a, #Use the clustering proportions as starting values for the weigh
                      mom.start = T ) #Find starting values for shape and rate parameters  using method o
```

```
## number of iterations= 574
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
```

```
#Obtain estimations
Phi_gamma <-list(alpha = fit_gamma$gamma.pars[1,],
                 beta = fit_gamma$gamma.pars[2,],
                    a = fit_gamma$lambda)
```

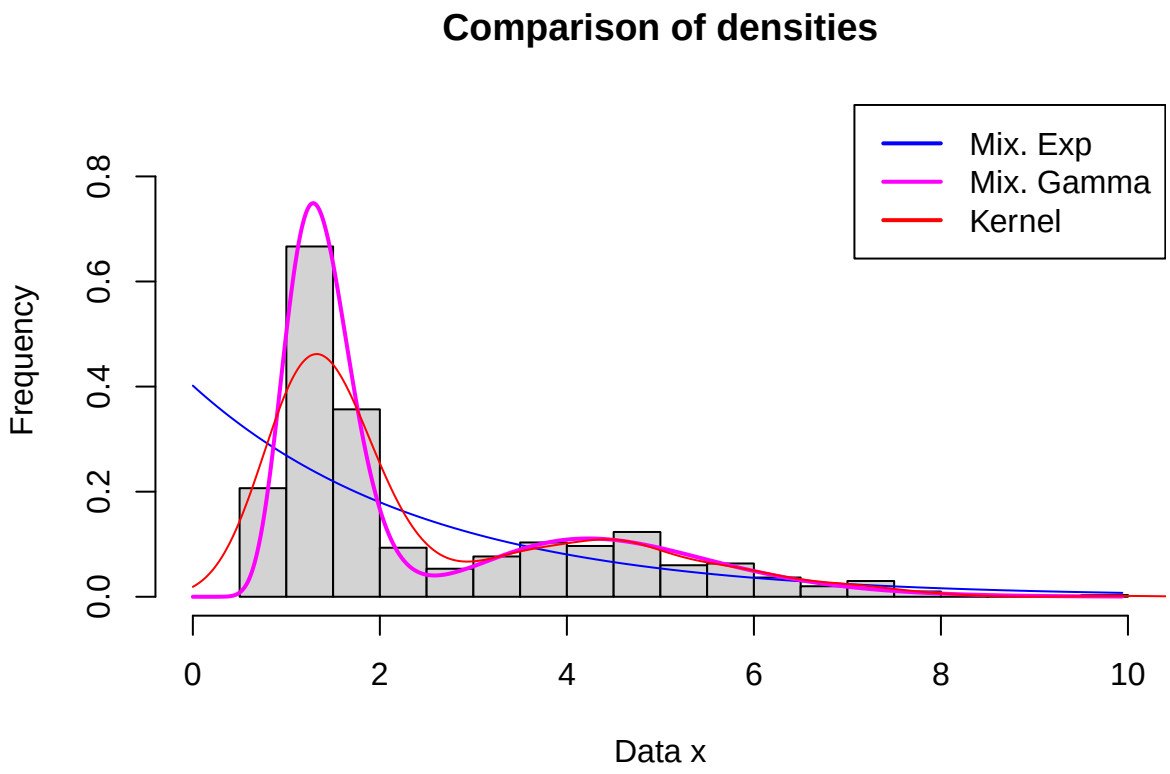
```r
#See estimates
Phi_gamma
```

```
## $alpha
##   comp.1   comp.2   comp.3   comp.4
## 14.60721 14.60721 14.60721 14.60721
##
## $beta
##     comp.1     comp.2     comp.3     comp.4
## 0.37860840 0.32555182 0.28531364 0.09456521
##
## $a
## [1] 0.04169407 0.17031933 0.12912047 0.65886614
```

Let's see how it fits the data:

```r
#Joint distribution
f_joint_gamma <- function(x,y,Phi){ Phi$a[y]*dgamma(x = x,
                                                    scale = Phi_gamma$beta[y],
                                                    shape = Phi_gamma$alpha[y] ) }

#Density of the mixture
fitted_density_gamma = function(x){ apply(X = outer(X = x,
                                                    Y = 1:m,
                                                    FUN = f_joint_gamma,
                                                    Phi=Phi_gamma ),
                                          MARGIN = 1,
                                          FUN = sum)}

#Some values for a grid
fval_gamma=fitted_density_gamma(xval)

#Plot the fitted densities
hist(x = X$x,   #Plot data
     main= "Comparison of densities",
     xlab = "Data x",
     ylab = "Frequency",
     breaks = 20,
     freq = F,
     ylim = c(0,0.9),
     xlim = c(0,10))
lines(xval,  #Plot fitted mixture of exponetial density
      fval_exp,
      type="l",
      col="blue")
lines(xval,  #Plot fitted mixture of gamma density
      fval_gamma,
      type="l",
      lwd=2,
      col="magenta")
lines(density(X$x,from = 0),  #Plot fitted kernel
      col="red")
legend('topright',
       col=c("blue","magenta", 'red'),
       lwd=2,
```

```
       legend=c("Mix. Exp", "Mix. Gamma", "Kernel"))
```



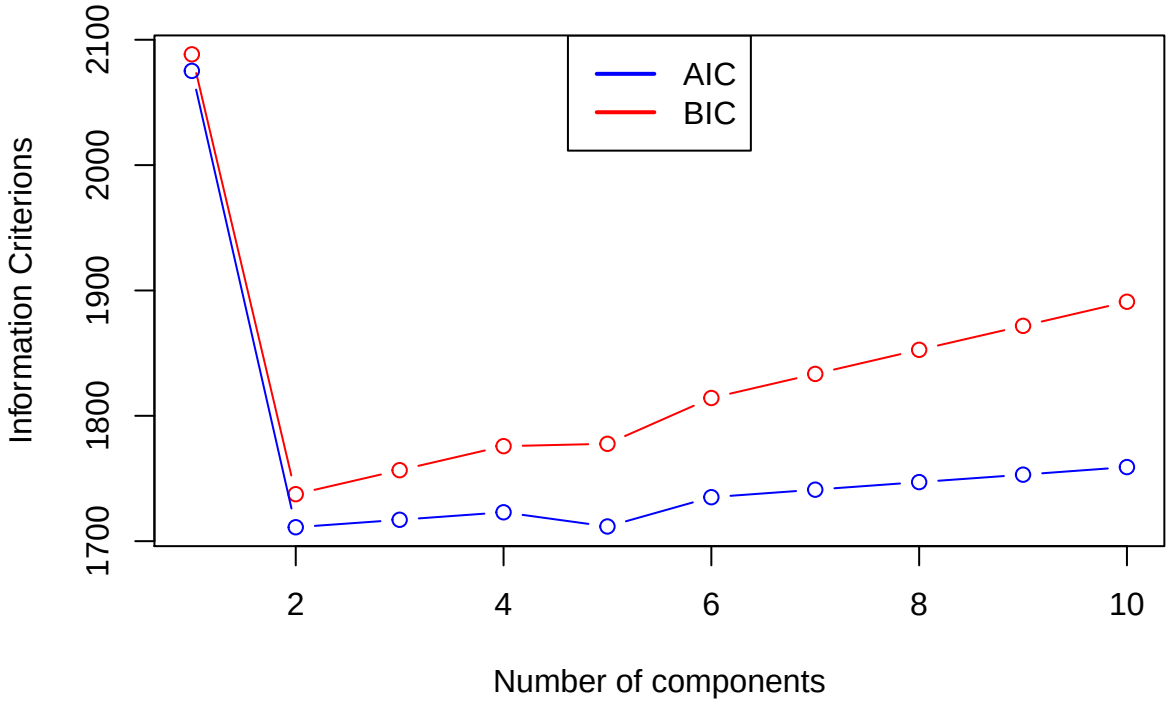
But how do we select the number of components?

There are several ways of doing this, the most common approach is using an information criterion, for instance the AIC (i.e $-2ln(L) + 2k$) or BIC (i.e $-2\log L + kln(n)$). In this setting, we fit the model with different valus for the number of componetns, and select the one that gives the minimum of the information criterion:

```
#Vector containing the values of information criteria
AIC=numeric(0)
BIC=numeric(0)

for(i in 1:10){ #Fit for m=1, ..., 10 componets. Here 10 was arbitrary.

  #Fit mixture model
  fit_gamma_temp<- gammamixEM(x = X$x, k = i )

  #Obtain information criterions
  AIC_temp <- 2*(3*i)-2*(fit_gamma_temp$loglik)
  BIC_temp <- log(n)*(3*i)-2*(fit_gamma_temp$loglik)

  #Store values in the vector
  AIC <-c(AIC,AIC_temp)
  BIC <-c(BIC,BIC_temp)

}
```

```
## number of iterations= 49
## number of iterations= 19
## number of iterations= 487
## number of iterations= 229
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
## number of iterations= 912
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
## number of iterations= 1031
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
## number of iterations= 944
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
## number of iterations= 1456
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
## One of the variances is going to zero;  trying new starting values.
## number of iterations= 486
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
## number of iterations= 1891
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
## One of the variances is going to zero;  trying new starting values.
## number of iterations= 1614
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
```

```r
#Plot AIC and BIC curves
plot(1:length(BIC),BIC,
     xlab = "Number of components",
     ylab = "Information Criterions",
     type = "b",
     col="red",
     ylim = c(min(AIC),max(BIC)))
lines(1:length(AIC),AIC,
      type = "b",
      col="blue")
legend('top',
       col=c("blue", 'red'),
       lwd=2,
       legend=c("AIC", "BIC"))
```



Therefore the AIC indicates that we should use 2 components, and the BIC 2. With that said the model to use is:

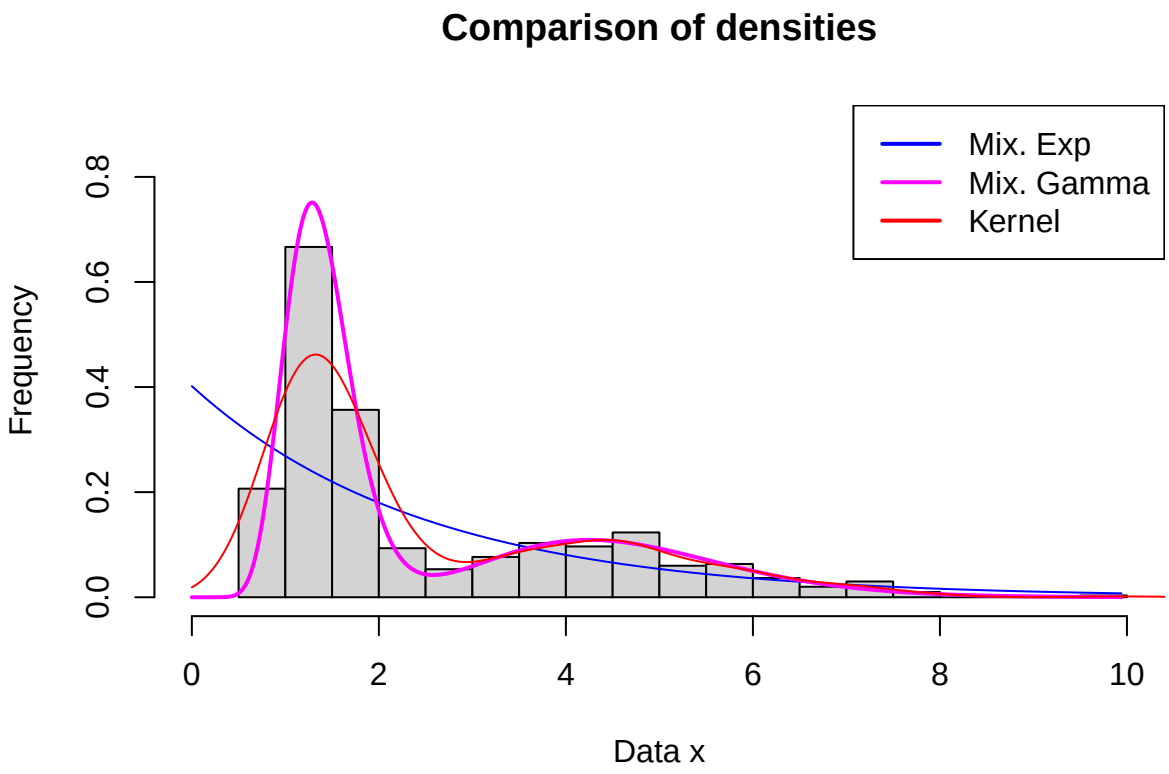
```
#Fit the model with the number of components that minimizes AIC
fit_gamma<- gammamixEM(x = X$x, #Data
                       k = which.min(AIC))
```

```
## number of iterations= 24
## number of iterations= 487
```

```
#Obtain estimations
Phi_gamma <-list(alpha = fit_gamma$gamma.pars[1,],
                 beta = fit_gamma$gamma.pars[2,],
                    a = fit_gamma$lambda)

#See estimates
Phi_gamma
```

```
## $alpha
##   comp.1   comp.2
## 14.72768 12.64119
##
## $beta
##     comp.1     comp.2
## 0.09364257 0.36498339
##
## $a
## [1] 0.6569604 0.3430396
```

```r
#Density of the mixture
fitted_density_gamma = function(x){ apply(X = outer(X = x,
                                                    Y = 1:which.min(AIC),
                                                    FUN = f_joint_gamma,
                                                    Phi=Phi_gamma ),
                                          MARGIN = 1,
                                          FUN = sum)}

#Some values for a grid
fval_gamma=fitted_density_gamma(xval)


#Plot the fitted densities
hist(x = X$x,   #Plot data
     main= "Comparison of densities",
     xlab = "Data x",
     ylab = "Frequency",
     breaks = 20,
     freq = F,
     ylim = c(0,0.9),
     xlim = c(0,10))
lines(xval,  #Plot fitted mixture of exponetial density
      fval_exp,
      type="l",
      col="blue")
lines(xval,  #Plot fitted mixture of gamma density
      fval_gamma,
      type="l",
      lwd=2,
      col="magenta")
lines(density(X$x,from = 0),  #Plot fitted kernel
      col="red")
legend('topright',
       col=c("blue","magenta", 'red'),
       lwd=2,
       legend=c("Mix. Exp", "Mix. Gamma", "Kernel"))
```


Comparison of densities
Mix. Exp
Mix. Gamma
Kernel
Frequency
0.0
0.2
0.4
0.6
0.8
0
2
4
6
8
10
Data x