Homework 4

Davide Berasi

2023-07-05

Exercise 2: Mixture of binomials

In the below table there are 40 observations $y_i \sim Bin(m_i, p_i)$. Data are also available in the file homework-bs-4-1.Rdata (http://datascience.maths.unitn.it/~claudio/teaching/bs/2022/homework/homework-bs-4-1.Rdata)

m	28	35	20	35	32	25	33	38	39	34	28	37	27	27	26	31	23	39	28	33	32	31	31	32	28	38	38	24	29	23	30	27	27	35	35	31	27	35
y	8	33	7	11	11	2	30	36	10	11	8	3	24	6	5	26	1	35	27	32	6	26	25	0	2	9	4	8	2	3	3	24	6	10	3	29	8	6

- Assume that the number of distinct p_is is 4. Implement in R a Gibbs sampler based on data augmentation following the model discussed in class and available in the notes Ch-MCMC.pdf (http://datascience.maths.unitn.it/~claudio/teaching/bs/2022/pdfCh-MCMC.pdf)

In our model we suppose that $p_i \in \{\gamma_1, \dots, \gamma_k\} \subset [0,1]^k$ where k is the number of components we suppose there exist, in our case we know that $k = 4$. We consider the p_i as independent random variables with $\mathbb{P}(p_i = \gamma_j) = \alpha_j$, where $\alpha = (\alpha_1, \dots, \alpha_k) \in \Delta^{k-1}$. In the Bayesian framework α and γ are random variables and we assume they are independent. For them we chose the following prior distributions:

$$\gamma_j \overset{\text{ind}}{\sim} Beta(a_j, b_j), \quad j = 1, \dots, k$$
$$\alpha \sim Dirichlet(c_1, \dots, c_k)$$

We are interested in the joint posterior distribution $\pi(\alpha, \gamma) = f(\alpha, \gamma | y) \propto f(y|\alpha, \gamma) f(\alpha, \gamma)$. This distribution can be written explicitly, but it is not easy to directly sample from it.

To simplify the problem we can use the data augmentation technique, that simply consists in introducing some new variables in such a way that the full conditional distributions can be computed easily.

In our case we define the categorical variable $C_i = j$ iff $p_i = \gamma_j$. Moreover it is useful to represent this variable also with its "one-hot encoding" z_{ij}, that is $z_{ij} = 1$ if $C_i = j$ and 0 otherwise.

Notice that the rows in the matrix $Z = (z_{ij})_{ij}$ are independent and that $\mathbb{P}(z_{ij} = 1 | \alpha, \beta) = \alpha_j$. Now we are interested in the joint posterior $\pi(\alpha, \gamma, Z) = f(\alpha, \gamma, Z | y) \propto f(y|\alpha, \gamma, Z) f(Z|\alpha, \gamma) f(\alpha, \gamma)$.

It is easy to verify that the full conditional posteriors are:

- $\alpha | (\gamma, Z, y) \sim Dirichlet(c_1 + z_{+1}, \dots, c_k + z_{+k})$ where $z_{+j} = \sum_{i=1}^k z_{ij}$
- $\gamma_j | (\alpha, Z, y) \sim Beta(a_j + y_{+j}, b_j + m_{+j} - y_{+j})$ where $y_{+j} = \sum_{i=1}^k z_{ij} y_i$ and $m_{+j} = \sum_{i=1}^k z_{ij} m_i$
- $\mathbb{P}(z_{ij} = 1 | \alpha, \beta) \propto \alpha_j \gamma_j^{y_i} (1 - \gamma_j)^{m_i - y_i}$

We have everything to implement our Gibbs sampler. In the implementation I prefer to work with the C_i instead of Z.

```
library("LearnBayes")

Gibbs <- function(n.iter, start, params, data, burn.in){
  m <- data$m
  y <- data$y
  k <- params$k
  n <- length(y)

  alpha.seq <- matrix(NA, ncol=k, nrow=n.iter)
  gamma.seq <- matrix(NA, ncol=k, nrow=n.iter)
  C.seq <- matrix(NA, ncol=n, nrow=n.iter)


  C <- C.seq[1,] <- start$C
  alpha <- alpha.seq[1,] <- start$alpha
  gamma <- gamma.seq[1,] <- start$gamma

  for(l in 2:n.iter){
    # alpha | gamma, Z
    z.plus_j <- tabulate(C, nbins = k)
    alpha <- rdirichlet(1, params$c + z.plus_j)

    # gamma | alpha, Z
    y.plus_j <- apply(as.matrix(1:k), 1, function(j) sum(y[C==j]))
    m.plus_j <- apply(as.matrix(1:k), 1, function(j) sum(m[C==j]))
    gamma <- rbeta(k, params$a + y.plus_j, params$b + m.plus_j - y.plus_j)
    gamma <- sort(gamma) # To avoid label switching

    # C | alpha, gamma
    for (i in 1:n){
      prob <- alpha * gamma^y[i] * (1 - gamma)^(m[i]-y[i])
      prob <- prob / sum(prob)
      C[i] <- sample(1:k, 1, prob=prob)
    }

    alpha.seq[l,] <- alpha
    gamma.seq[l,] <- gamma
    C.seq[l,] <- C
  }
  result <- list(alpha = alpha.seq[-(1:burn.in),],
                 gamma = gamma.seq[-(1:burn.in),],
                 C = C.seq[-(1:burn.in),])
  return(result)
}
```

- Run the Gibbs sampler for the above data and summarize your finding.

As initial state we order the data according to $p = y/m$ and split them in the 4 quarters, for α we take a uniform distribution and for γ we take the means of p in the 4 initial components.

As hyperparameters we take $a_j = b_j = c_j = 1$ for each $j = 1, \dots, k$, so that the prior for α and γ are uniform distributions.

With this values we run the chain for 5000 iterations and we discard the first 1000.

```
k <- 4
p <- y/m
sorted.idx <- order(p)
y <- y[sorted.idx]
m <- m[sorted.idx]
quarters <- rep(1:4, each=10)

start <- list(alpha = rep(1/k, k),
              gamma = apply(as.matrix(1:k), 1, function(j) mean(p[quarters==j])),
              C = quarters)

params <- list(c = rep(1, k),
               a = rep(1, k), b = rep(1, k),
               k = k)

system.time(result <- Gibbs(5000, start, params, data=list(m=m, y=y), burn.in = 1000))
```

```
##    utente   sistema trascorso
##      2.43      0.03      2.69
```

As estimators for α and γ we take the the median of the corresponding posterior, for C_i is the closest integer to the median of the posterior.

It is interesting to see the values of our estimates for γ and for the number of observations belonging to each cluster.

```
C <- apply(result$C, 2, median)
gamma <- apply(result$gamma, 2, median)
alpha <- apply(result$alpha, 2, median)
N_j <- tabulate(C, nbins = k) # N_J[j] is the number of observations in cluster j. It is equal to z.plus_j
rbind(N_j, gamma)
```

```
##             [,1]       [,2]      [,3]       [,4]
## N_j   9.00000000 13.0000000 6.0000000 12.0000000
## gamma 0.08401411  0.2319378 0.3021896  0.9051009
```

We actually know the true value of γ, and we can see that the distance of our estimate from it is small.

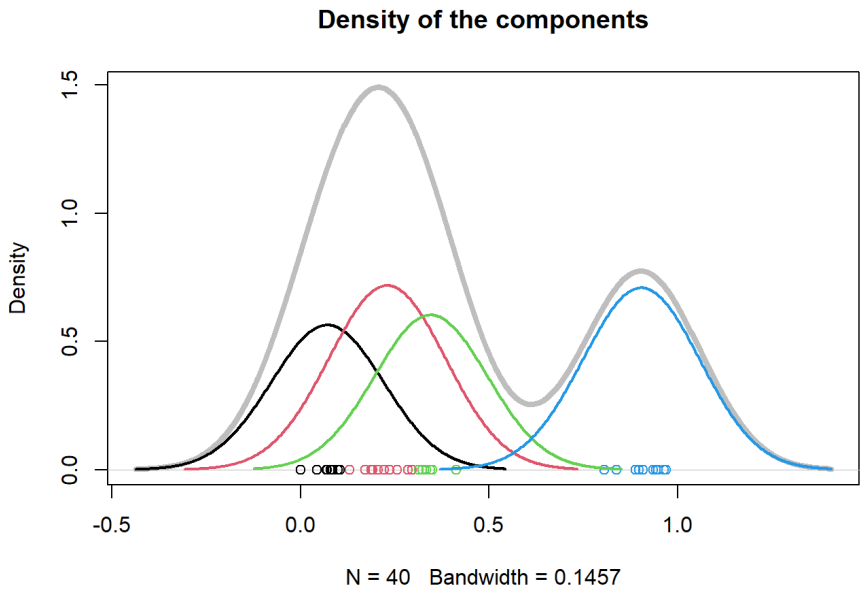
```
norm(gamma - sort(theta), "2")
```

```
## [1] 0.05693379
```

It is interesting to compare the density of y_i/m_i with the densities of the 4 components. We can notice that 3 of the clusters are close and they overlap.

```
# Helper function to plot the density of each component.
plot.clusters.densities <- function(y, C, alpha){
  k <- length(alpha)
  bw <- bw.nrd0(y)

  plot(density(y, bw=bw), lwd=4, col="gray", main = "Density of the components" )
  points(cbind(y,0), col=C)
  for (j in 1:k) {
    if (sum(C==j) > 0){
      d <- density(y[C==j], bw=bw)
      d$y <- d$y * alpha[j]
      lines(d, col=j, lwd=2)
    }
  }
}
```

```
plot.clusters.densities(y/m, C, alpha)
```



- Use JAGS to implement the same model and compare the results

We define the model in BUGS syntax

```
library(R2jags)
```

```
## Caricamento del pacchetto richiesto: rjags
```

```
## Caricamento del pacchetto richiesto: coda
```

```
## Linked to JAGS 4.3.1
```

```
## Loaded modules: basemod,bugs
```

```
##
## Caricamento pacchetto: 'R2jags'
```

```
## Il seguente oggetto è mascherato da 'package:coda':
##
##     traceplot
```

```
mixture <- "model {
  a <- c(1,1,1,1)
  b <- c(1,1,1,1)
  c <- c(1,1,1,1)

  # Likelihood
  for (i in 1:N) {
    y[i] ~ dbin(p[i], m[i])
    p[i] <- gamma[C[i]]
    C[i] ~ dcat(alpha[1:k])
  }

  # Priors
  alpha ~ ddirch(c[1:k])
  for (j in 1:k) {
    gamma_aux[j] ~ dbeta(a[j], b[j])
  }
  gamma[1:k] <- sort(gamma_aux[1:k])
}"
```

and we run the chain for 5000 iterations.

```
data <- list(y=y, m=m, N=40, k=k)
parameters <- c("alpha", "gamma", "C")

mixturerun <- jags(
  data = data,
  parameters.to.save = parameters,
  model.file = textConnection(mixture),
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 1000
)
```

```
## module glm loaded
```

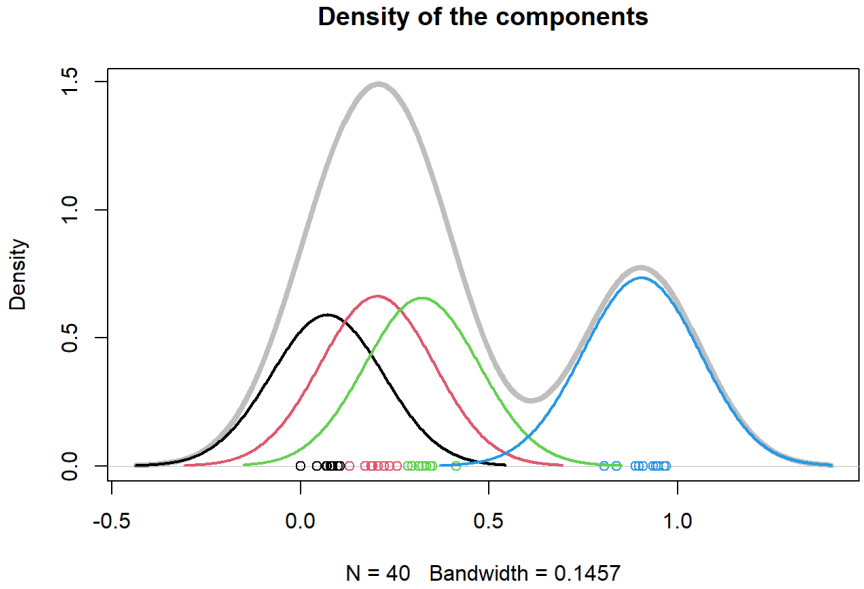
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 40
##    Unobserved stochastic nodes: 45
##    Total graph size: 183
##
## Initializing model
```

The results are quite similar to what we obtained with our Gibbs sampler.

```
C <- round(mixturerun$BUGSoutput$median$C)
alpha <- mixturerun$BUGSoutput$median$alpha
gamma <- mixturerun$BUGSoutput$median$gamma

N_j <- tabulate(C, nbins=4)
rbind(N_j, gamma)
```

```
##            [,1]      [,2]       [,3]       [,4]
## N_j   9.00000000 9.0000000 10.0000000 12.0000000
## gamma 0.08363648 0.2350055  0.2959505  0.9048819
```

```
plot.clusters.densities(y/m, C, alpha)
```



- Assume now the number of components (distinct p_i) is unknown. Implement in R a non parametric Gibbs sampler with stick-breaking approach and Dirichlet prior

In the non-parametric setup our model is:

$$y_i \,|\, C_i, \gamma \sim Bin(m_i, p_{C_i})$$
$$C_i \,|\, \pi \sim \pi, \;\; \pi \sim GEM(\beta)$$
$$p_j \sim Beta(a, b)$$

Where $GEM(\beta)$ is the random measure $\sum_{j=1}^{\infty} \pi_j \delta_j$ where the weights π_j are generated with the stick-breaking process:

$$\pi_j := \mathbb{P}(\pi = j) = \gamma_j \prod_{i=i}^{j-1}(1 - \gamma_i)$$
$$\gamma_i \overset{\text{ind}}{\sim} Gamma(\alpha, \beta)$$

Following this notes (http://datascience.maths.unitn.it/~claudio/teaching/bs/2022/pdf/lec1.pdf) I implement the blocked Gibbs sampler. In this algorithm the π_j are treated as parameters and their posterior given the C_i is still a stick-breacking process, where

$$\gamma_i \overset{\text{ind}}{\sim} Gamma\left(\alpha + N_i, \; \beta + \sum_{s=i+1}^{k} N_s\right)$$

```
Gibbs.non_param <- function(n.iter, start, params, data, burn.in){
  m <- data$m
  y <- data$y
  n <- length(y)
  k <- params$k  # Maximum number of clusters

  C.seq <- matrix(NA, ncol=n, nrow=n.iter)
  p.seq <- matrix(NA, ncol=k, nrow=n.iter)
  N_j.seq <- matrix(NA, ncol=k, nrow=n.iter)
  PI.seq <- matrix(NA, ncol=k, nrow=n.iter)

  C <- C.seq[1,] <- start$C
  p <- p.seq[1,] <- start$p
  PI <- PI.seq[1,] <- start$PI
  N_j <- N_j.seq[1,] <- tabulate(C, nbins = k)

  for(l in 2:n.iter){
    # p_i | Z_i, y_i
    y.plus_j <- apply(as.matrix(1:k), 1, function(j) sum(y[C==j]))
    m.plus_j <- apply(as.matrix(1:k), 1, function(j) sum(m[C==j]))
    p <- rbeta(k, params$a + y.plus_j, params$b + m.plus_j-y.plus_j)

    # PI | Z_i
    stick.length <- 1
    for (j in 1:(k-1)){
      PI[j] <- stick.length * rbeta(1, params$alpha + N_j[j], params$beta + sum(N_j[(j+1):k]))
      stick.length <- stick.length - PI[j]
    }
    PI[k] <- 1 - sum(PI[1:(k-1)])

    # Z_i | p_i, PI
    for (i in 1:n){
      prob <- PI * dbinom(y[i], m[i], prob=p)
      C[i] <- sample(1:k, 1, prob = prob)
    }
    # Update N_j
    N_j <- tabulate(C, nbins = k)

    C.seq[l,] <- C
    PI.seq[l,] <- PI
    p.seq[l,] <- p
    N_j.seq[l,] <- N_j
  }

  result <- list(p = p.seq[-(1:burn.in),],
                 C = C.seq[-(1:burn.in),],
                 N_j = N_j.seq[-(1:burn.in),],
                 PI = PI.seq[-(1:burn.in),])
  return(result)
}
```

- Run the non parametric Gibbs sampler for the above data and summarize your finding. In particular discuss the posterior distribution of the number of clusters

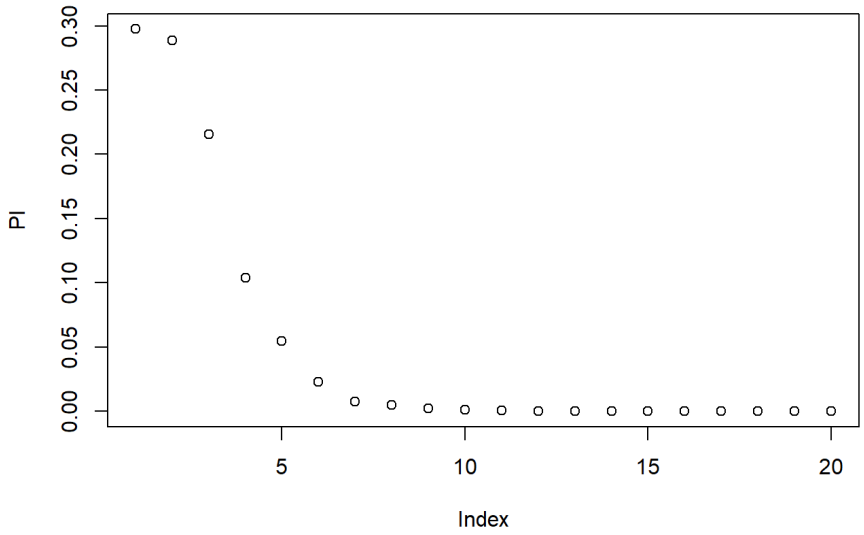
Now we run the algorithm and we study the number of clusters.

```
k <- 20
p <- y/m
sorted.idx <- sort.int(p, index.return = TRUE)$ix
y <- y[sorted.idx]
m <- m[sorted.idx]
components <- rep(1:k, each=2)

start <- list(PI = rep(1/k, k),
              p = apply(as.matrix(1:k), 1, function(j) mean(p[components==j])),
              C = components)

params <- list(a = 1, b = 1,
               alpha = 1, beta = 1,
               k = k)

system.time(result <- Gibbs.non_param(5000, start, params, data=list(m=m, y=y), burn.in = 1000))
```

```
##    utente   sistema trascorso
##      4.00      0.00      4.42
```

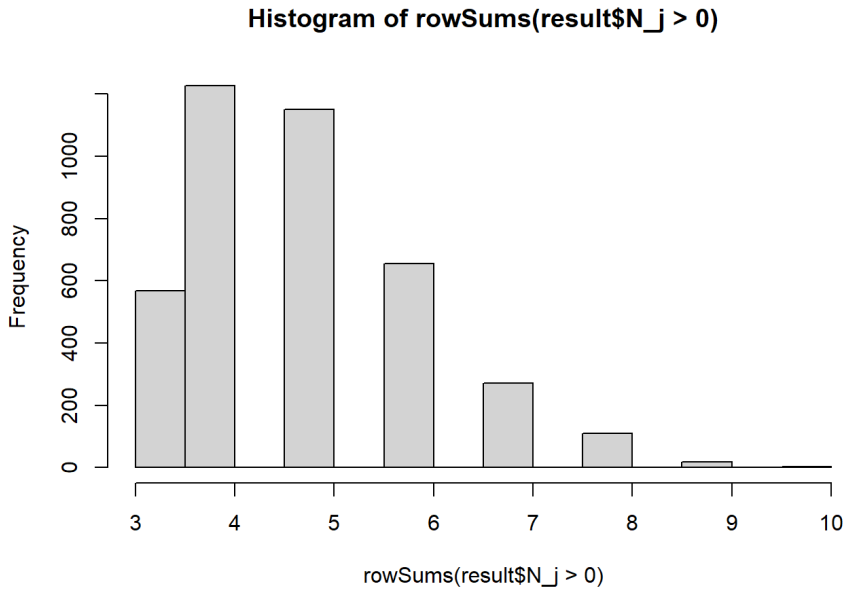
We can notice that,on average the π_j are close to 0 for $j > 5$. This indicates that probably there are no more than 5 components.

```
PI <- colMeans(result$PI)
N_j <- tabulate(C, k)

plot(PI)
```



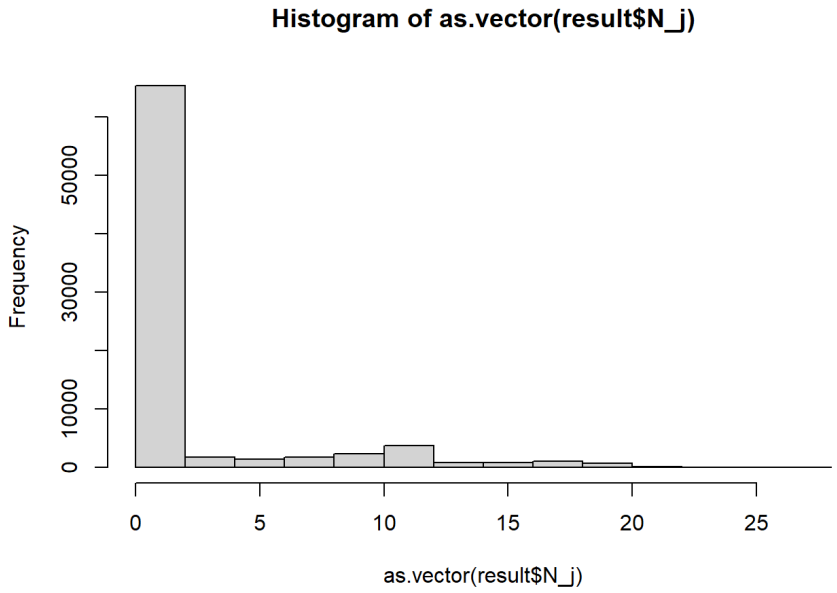
To confirm that, we can look also at the distribution of the number of clusters.

```
hist(rowSums(result$N_j>0))
```



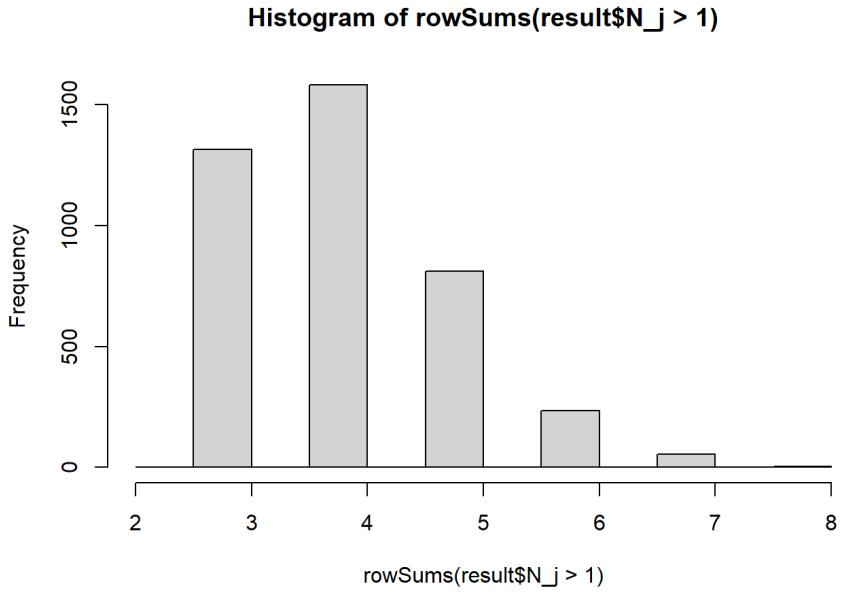
We can notice that a lot of the cluster we are considering have few components.

```
hist(as.vector(result$N_j))
```



So, it makes more sense to not consider the clusters with a small number of elements. In this case it is even more evident that the number of components should be 4.

```
hist(rowSums(result$N_j>1))
```



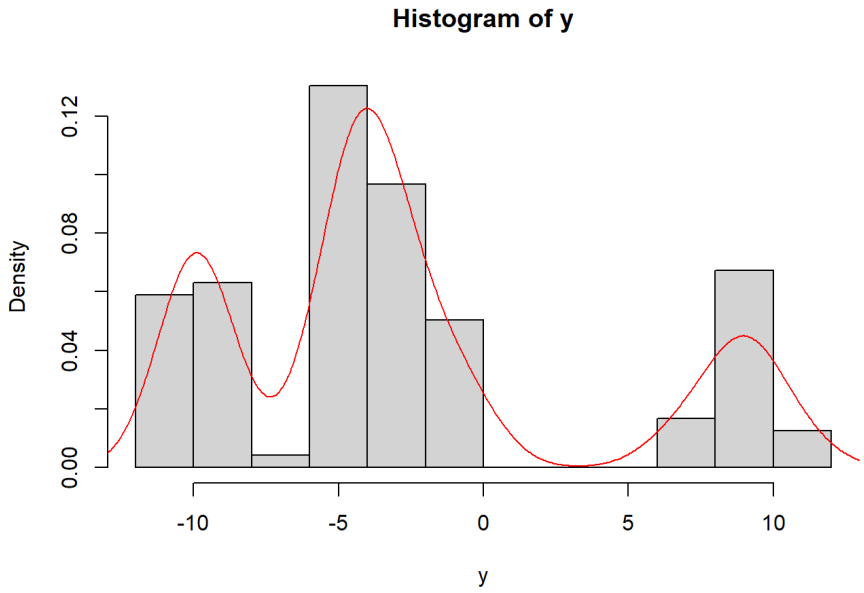
Exercise 1: Mixture of Gaussians

In the file homework-bs-4-2.Rdata (http://datascience.maths.unitn.it/~claudio/teaching/bs/2022/homework/homework-bs-4-2.Rdata) there are measurements from a Gaussian mixture where each of the component of the mixture is a univariate normal distribution with unknown mean and variance

- By graphical inspection guess the number of components and implement in R a Gibbs sampler based on data augmentation following the model discussed in the notes Ch-MCMC.pdf (http://datascience.maths.unitn.it/~claudio/teaching/bs/2022/pdfCh-MCMC.pdf)

From the plot of the histogram it look like the data comes from 3 different groups.

```
hist(y, freq = FALSE)
lines(density(y), col="red")
```



We have $n = 119$ observations and we model them as a mixture of $k = 3$ gaussian. As in the previous exercise, we consider the categorical variables C_i (and their one-hot encoding representation Z) that tells us the component of observation i, that is $C_i = j$ iff observation i belongs to component j.

To construct the model I also followed this notes (https://web.uniroma1.it/memotef/sites/default/files/file%20lezioni/Lezione3_CMollica.pdf).

Defined $\mu = (\mu_1, \dots, \mu_k)$ and $\tau = (\tau_1, \dots, \tau_k)$, our model is:

$$y_i \,|\, (C_i, \mu, \tau^2) \sim \mathcal{N}(\mu_{C_i}, \tau_{C_i}^{-2}), \quad i = 1, \dots, n$$
$$C_i \,|\, \alpha \sim Cat(\alpha), \quad i = 1, \dots, n$$

with prior distributions

$$\alpha \sim Dirichlet(c), \quad c = (c_1, \dots, c_k)$$
$$\mu_j \overset{\text{ind}}{\sim} \mathcal{N}(\mu_0, \tau_0^{-2}), \quad j = 1, \dots, k$$
$$\tau_j \overset{\text{ind}}{\sim} Gamma(a, b), \quad j = 1, \dots, k$$

where a, b, c, μ_0, τ_0 are fixed hyperparameters.

We are interested in the joint posterior distribution $\pi(\alpha, \mu, \tau, Z)$. It can be written explicitly and from it we obtain the full conditional distributions.

- $\alpha | (\mu, \tau, Z) \sim Dirichlet(c_1 + z_{+1}, \dots, c_k + z_{+k})$
- $\mu_j \,|\, (\alpha, \tau, Z) \sim \mathcal{N}\left(\delta_j \frac{y_{+j}}{z_{+j}} + (1 - \delta_j)\mu_0,\ (\tau_0 + z_{+j}\tau_j)^{-2}\right)$
- $\tau_j \,|\, (\alpha, \mu, Z) \sim Gamma(a + z_{+j}/2, b + SS_j/2)$
- $\mathbb{P}(z_{ij} = 1 | \alpha, \mu, \tau) \propto \alpha_j \mathcal{N}(y_i | \mu_j, \tau_j^{-2})$

where

$$SS_j = \sum_{i=1}^{n} z_{ij}(y_i - \mu_j)^2 \qquad \delta_j = \frac{z_{+j}\tau_j}{\tau_0 + z_{+j}\tau_j}$$

We have everything to implement our Gibbs sampler.

```
Gibbs2 <- function(n.iter, start, params, data, burn.in){
  y <- data
  k <- params$k
  n <- length(y)

  alpha.seq <- matrix(NA, ncol=k, nrow=n.iter)
  mu.seq <- matrix(NA, ncol=k, nrow=n.iter)
  A.seq <- matrix(NA, ncol=k, nrow=n.iter)
  C.seq <- matrix(NA, ncol=n, nrow=n.iter)

  alpha <- alpha.seq[1,] <- start$alpha
  mu <- mu.seq[1,] <- start$mu
  A <- A.seq[1,] <- 1 / start$A # It is tau
  C <- C.seq[1,] <- start$C

  for(l in 2:n.iter){
    # alpha | mu, A, Z
    z.plus_j <- tabulate(C, nbins = k)
    alpha <- rdirichlet(1, params$c + z.plus_j)

    # A | alpha, mu, Z
    SS_j <- apply(as.matrix(1:k), 1, function(j) sum( ((y - mu[j])^2)[C==j]) )
    A <- rgamma(k, shape=params$a + z.plus_j/2,
                   rate=params$b + SS_j/2)

    # mu | alpha, A, Z
    y.plus_j <- apply(as.matrix(1:k), 1, function(j) sum(y[C==j]))
    mean.plus_j <- y.plus_j / z.plus_j
    mean.plus_j[y.plus_j==0] <- 0
    delta_j <- z.plus_j * A / (params$tau0 + z.plus_j*A)

    mu <- rnorm(k,
                mean = delta_j * mean.plus_j + (1 - delta_j) * params$mu0,
                sd = sqrt(1 / (A * z.plus_j + params$tau0)))
    mu <- sort(mu) # Avoid label switching

    # Z | alpha, mu, A
    for (i in 1:n){
      prob <- alpha * dnorm(y[i], mu, sqrt(1/A))
      prob <- prob / sum(prob)
      C[i] <- sample(1:k, 1, prob = prob)
    }

    alpha.seq[l,] <- alpha
    mu.seq[l,] <- mu
    A.seq[l,] <- A
    C.seq[l,] <- C

  }
  result <- list(alpha = alpha.seq[-(1:burn.in),],
                 mu = mu.seq[-(1:burn.in),],
                 A = A.seq[-(1:burn.in),],
                 C = C.seq[-(1:burn.in),])
  return( result )
}
```

- Run the Gibbs sampler for the above data and summarize your finding

```
k <- 3
y <- sort(y)
thirds <- rep(1:k, each=length(y)/k+1)[1:length(y)]
start <- list(alpha = rep(1/k, k),
              mu = apply(as.matrix(1:k), 1, function(j) mean(y[thirds==j])),
              A = 1 / apply(as.matrix(1:k), 1, function(j) var(y[thirds==j])),
              C = thirds)

params <- list(c = rep(1, k),
              mu0 = mean(y),
              tau0 = 0.01,
              a = 1, b = 1,
              k=k)
system.time(result <- Gibbs2(5000, start, params, y, burn.in=1000))
```

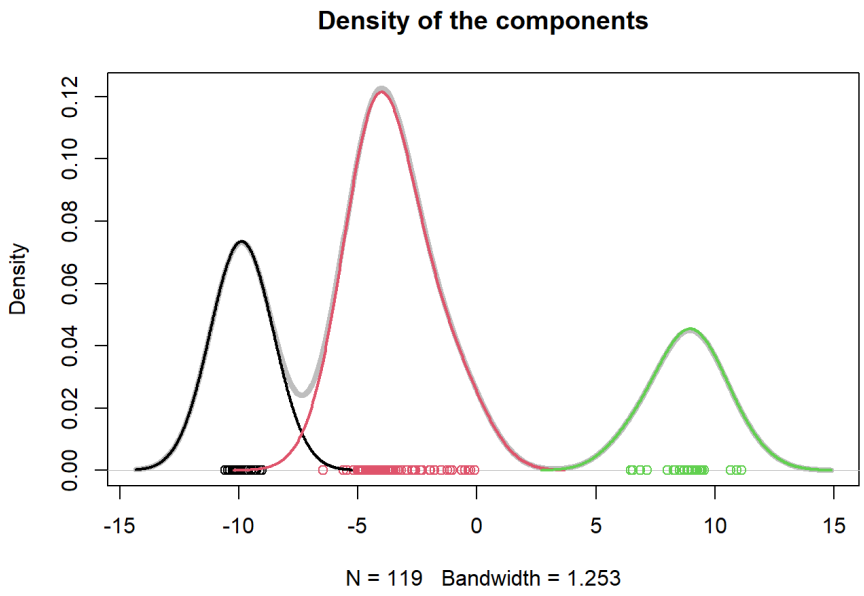
```
##    utente   sistema trascorso
##      7.02      0.00      8.02
```

As before, for each component we can estimate $\mu_j, \sigma_j^2 = \tau_j^{-2}$ and the number of points in each cluster.

```
C <- apply(result$C, 2, median)
mu <- apply(result$mu, 2, median)
sigma2 <- 1 / apply(result$A, 2, median)
alpha <- apply(result$alpha, 2, median)

N_j <- tabulate(C, nbins = k)
rbind(mu, sigma2, N_j)
```

```
##             [,1]      [,2]      [,3]
## mu     -9.888677 -3.534692  8.829074
## sigma2  0.254888  2.112397  1.531611
## N_j    29.000000 67.000000 23.000000
```

```
plot.clusters.densities(y, C, alpha)
```



- Use JAGS to implement the same model and compare the results

We write the model in BUGS syntax

```
mixture2 <- "model {
  # hyperparameter
  a <- 1
  b <- 1
  c <- rep(1, k)
  mu0 <- 12.95 # mean(y)
  tau0 <- 1 / 100

  # Likelihood
  for (i in 1:N) {
    y[i] ~ dnorm(mu[C[i]], A[C[i]])
    C[i] ~ dcat(alpha[1:k])
  }

  # Priors
  alpha[1:k] ~ ddirich(c)
  for (j in 1:k){
    A[j] ~ dgamma(a, b)
    mu[j] ~ dnorm(mu0, tau0)
  }
}
"
```

and we run a chain for 5000 iterations

```
data <- list(y=y, N=length(y), k=3)
parameters <- c("alpha", "C", "mu", "A")

jagsfit2.1 <- jags(
  data = data,
  parameters.to.save = parameters,
  model.file = textConnection(mixture2),
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 1000
)
```

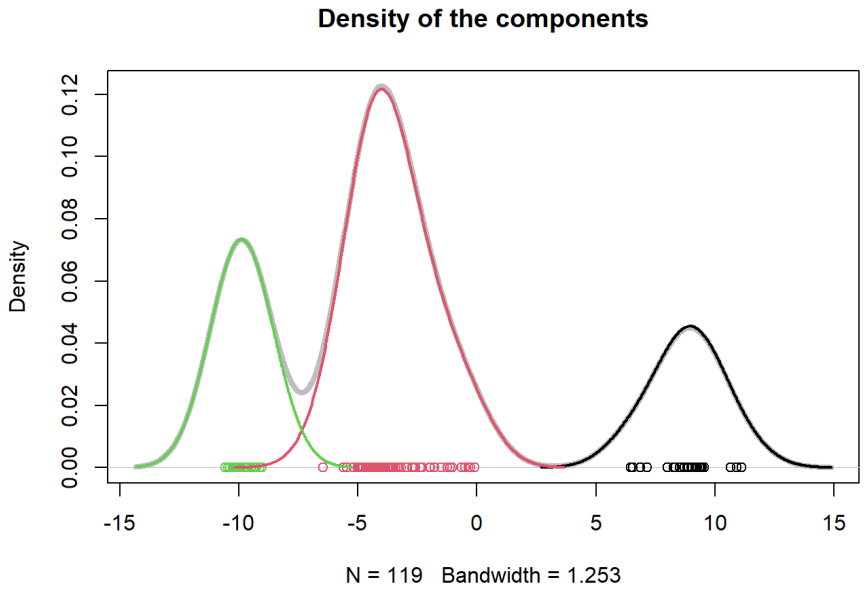
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 126
##    Total graph size: 490
##
## Initializing model
```

The results we obtain are almost identical.

```
C.1 <- round(jagsfit2.1$BUGSoutput$median$C)
mu.1 <- jagsfit2.1$BUGSoutput$median$mu
sigma2.1 <- 1 / jagsfit2.1$BUGSoutput$median$A
alpha.1 <- jagsfit2.1$BUGSoutput$median$alpha

N_j <- apply(as.matrix(1:k), 1, function(j) sum(C.1==j))
rbind(mu.1, sigma2.1, N_j)
```

```
##              [,1]      [,2]       [,3]
## mu.1      8.833013 -3.530009 -9.8837441
## sigma2.1  1.539511  2.142150  0.2532459
## N_j      23.000000 67.000000 29.0000000
```

```
plot.clusters.densities(y, C.1, alpha.1)
```



- Assume the number of components is actually one more then the one you guess. Repeat the analysis using both your algorithm and JAGS

Using my Gibbs sampler we have the following result:

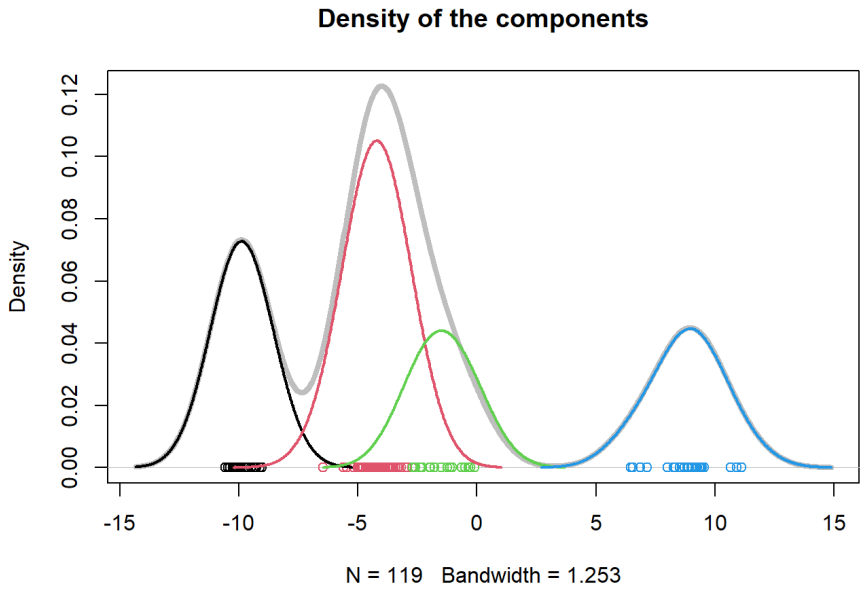
```
k <- 4
y <- sort(y)
quarters <- rep(1:k, each=length(y)/k+1)[1:length(y)]
start <- list(alpha = rep(1/k, k),
              mu = apply(as.matrix(1:k), 1, function(j) mean(y[quarters==j])),
              A = 1 / apply(as.matrix(1:k), 1, function(j) var(y[quarters==j])),
              C = quarters)

params <- list(c = rep(1, k),
              mu0 = mean(y),
              tau0 = 1 / 100,
              a = 1, b = 1,
              k = k)
system.time(result <- Gibbs2(5000, start, params, y, burn.in=1000))
```

```
##    utente   sistema trascorso
##      7.50      0.01      8.33
```

```
C <- apply(result$C, 2, median)
mu <- apply(result$mu, 2, median)
sigma2 <- 1 / apply(result$A, 2, median)
alpha <- apply(result$alpha, 2, median)

N_j <- apply(as.matrix(1:k), 1, function(j) sum(C==j))
rbind(mu, sigma2, N_j)
```

```
##              [,1]       [,2]      [,3]      [,4]
## mu     -9.8879087 -4.2071335 -1.846515  8.818650
## sigma2  0.2547442  0.6908156  1.507960  1.543753
## N_j    29.0000000 50.0000000 17.000000 23.000000
```

```
plot.clusters.densities(y, C, alpha)
```

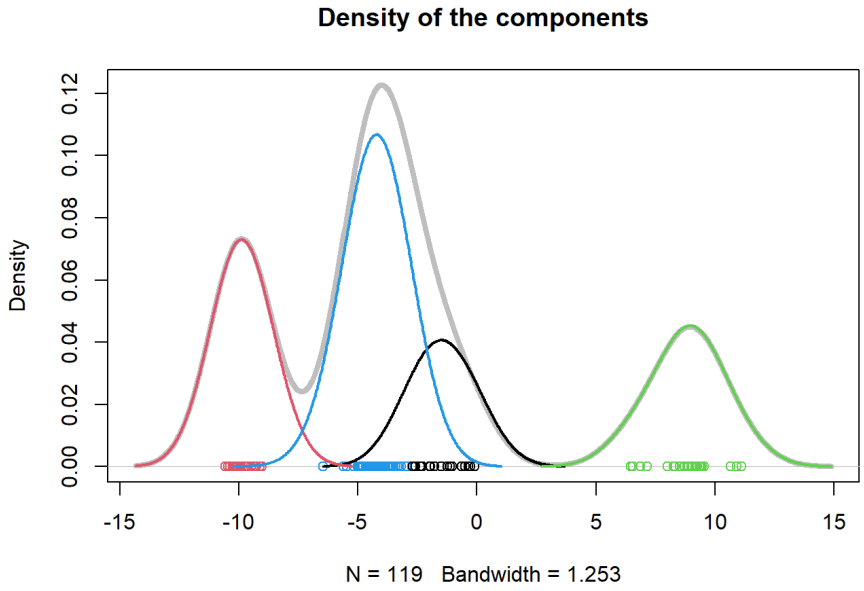


Using JAGS we obtain quite similar results.

```
data <- list(y=y, N=length(y), k=4)
parameters <- c("alpha", "C", "mu", "A")

jagsfit2.2 <- jags(
  data = data,
  parameters.to.save = parameters,
  model.file = textConnection(mixture2),
  n.chains = 1,
  n.iter = 5000,
  n.burnin = 1000
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 119
##    Unobserved stochastic nodes: 128
##    Total graph size: 492
##
## Initializing model
```

```
C.2 <- round(jagsfit2.2$BUGSoutput$median$C)
mu.2 <- jagsfit2.2$BUGSoutput$median$mu
alpha.2 <- jagsfit2.2$BUGSoutput$median$alpha

plot.clusters.densities(y, C.2, alpha.2)
```



- Suggest a way to compare the two models and propose a final model.

As an evaluation metric for the two models we can use the average silhouette coefficient.

Given the i-th observation, we define $a(i)$ as the average distance of i from the other point in its cluster

$$a(i) = \frac{1}{N_{C_i} - 1} \sum_{j \in C_i, i \neq j} |y_i - y_j| \qquad N_{C_i} = |\{\text{observations in componet } C_i\}|$$

and $b(i)$ as the average distance of i from the points in the closest cluster

$$b(i) = \min_{J \neq C_i} \frac{1}{N_J} \sum_{C_j = J} |y_i - y_j|$$

The silhouette coefficient of i is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{b(i),\, a(i)\}} \quad \text{if } N_{C_i} > 1, \;\; 0 \text{ otherwise}$$

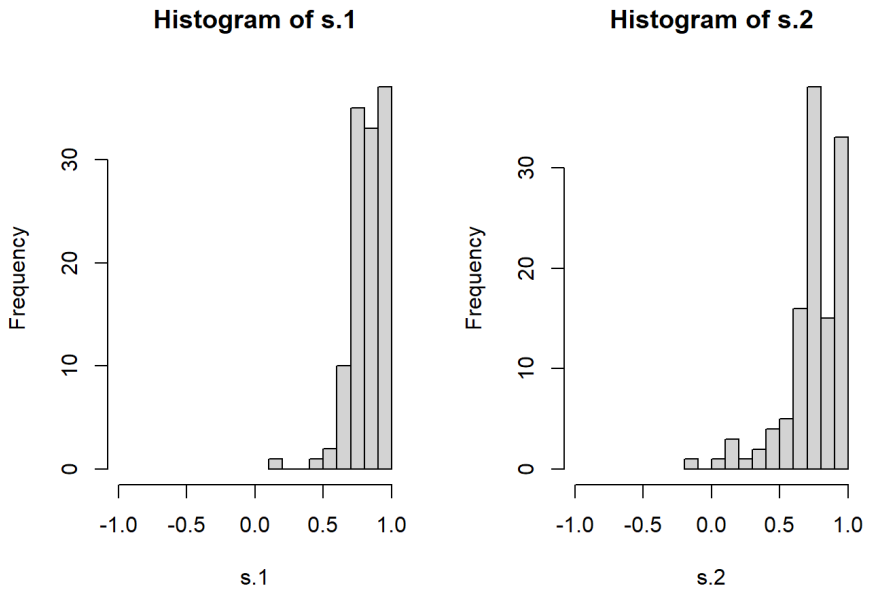
So $-1 \leq s(i) \leq 1$ and in general we say that observation i is well classified if $s(i)$ is close to one.

Here is a function that computes the silhouette coefficients for a given clustering of the data.

```
silhouette <- function(y, C, k){
  n <- length(y)
  a <- rep(NA, n)
  b <- rep(NA, n)
  s <- rep(0, n)
  for (i in 1:n){
    mask <- (C == C[i])
    if (sum(mask)>1){
      mask[i] <- FALSE
      a[i] <- mean( abs(y[i] - y[mask]) )

      J_ne_Ci <- as.matrix((1:k)[-C[i]])
      b[i] <- min(apply(J_ne_Ci, 1, function(j) mean(abs(y[i]-y[C==j])) ))
      s[i] <- (b[i] - a[i]) / max(a[i], b[i])
    }
  }
  return(s)
}
```

Looking at the histograms of the silhouette coefficients, we can see that in the second model some points are considered as missclassified and overall the clustering given by the first model looks better.

```
s.1 <- silhouette(y, C.1, k=3)
s.2 <- silhouette(y, C.2, k=4)

par(mfrow=c(1, 2))
hist(s.1, xlim = c(-1, 1))
hist(s.2, xlim = c(-1, 1))
```



Even though we know that the true number of clusters is 4, the mean silhouette coefficients metric tells us that the clustering with $k = 3$ is slightly better.

```
colMeans(cbind(s.1, s.2))
```

```
##       s.1       s.2
## 0.8112138 0.7439111
```

- Assume now the number of components is unknown. Implement in R a non parametric Gibbs sampler with stick-breaking approach and Dirichlet prior

As in previous exercise I use the blocked Gibbs sampler algorithm.

```
Gibbs2.non_param <- function(n.iter, start, params, data, burn.in){
  y <- data
  n <- length(y)
  k <- params$k  # Maximum number of clusters
  beta <- params$beta

  C.seq <- matrix(NA, ncol=n, nrow=n.iter)
  mu.seq <- matrix(NA, ncol=k, nrow=n.iter)
  A.seq <- matrix(NA, ncol=k, nrow=n.iter)
  N_j.seq <- matrix(NA, ncol=k, nrow=n.iter)
  PI.seq <- matrix(NA, ncol=k, nrow=n.iter)

  C <- C.seq[1,] <- start$C
  mu <- mu.seq[1,] <- start$mu
  A <- A.seq[1,] <- start$A
  PI <- PI.seq[1,] <- start$PI
  N_j <- N_j.seq[1,] <- tabulate(C, k)

  for(l in 2:n.iter){
    # mu | alpha, A, Z
    y.plus_j <- apply(as.matrix(1:k), 1, function(j) sum(y[C==j]))
    mean.plus_j <- y.plus_j / N_j
    mean.plus_j[y.plus_j==0] <- 0
    delta_j <- N_j * A / (params$tau0 + N_j*A)
    mu <- rnorm(k, mean = delta_j * mean.plus_j + (1 - delta_j) * params$mu0,
                sd = sqrt(1 / (A * N_j + params$tau0)))

    # A | alpha, mu, Z
    SS_j <- apply(as.matrix(1:k), 1, function(j) sum( ((y - mu[j])^2)[C==j]) )
    A <- rgamma(k, shape=params$a + N_j/2,
                   rate=params$b + SS_j/2)

    # Z_i | p_i, PI, X_i
    for (i in 1:n){
      prob <- PI * dnorm(y[i], mu, 1 / sqrt(A))
      C[i] <- sample(1:k, 1, prob = prob)
    }
    # Update N_j
    N_j <- tabulate(C, k)

    # PI | Z_i
    stick.length <- 1
    for (j in 1:(k-1)){
      PI[j] <- stick.length * rbeta(1, params$alpha + N_j[j], params$beta + sum(N_j[-(1:j)]))
      stick.length <- stick.length - PI[j]
    }
    PI[k] <- 1 - sum(PI[1:(k-1)])

    C.seq[l,] <- C
    mu.seq[l,] <- mu
    A.seq[l,] <- A
    N_j.seq[l,] <- N_j
    PI.seq[l,] <- PI
  }

  result <- list(mu = mu.seq[-(1:burn.in),],
                 A = A.seq[-(1:burn.in),],
                 C = C.seq[-(1:burn.in),],
                 N_j = N_j.seq[-(1:burn.in),],
                 PI = PI.seq[-(1:burn.in),])
  return(result)
}
```

- Run the non parametric Gibbs sampler for the above data and summarize your finding. In particular discuss the posterior distribution of the number of clusters

```
k <- 20
y <- sort(y)
components <- rep(1:k, each=length(y)/k+1)[1:length(y)]
start <- list(PI = rep(1/k, k),
              mu = apply(as.matrix(1:k), 1, function(j) mean(y[components==j])),
              A = 1 / apply(as.matrix(1:k), 1, function(j) var(y[components==j])),
              C = components)

params <- list(c=rep(1, k),
               mu0 = mean(y),
               tau0 = 1 / 100,
               a = 1, b=1,
               alpha = 1, beta=1,
               k = k)

system.time(result <- Gibbs2.non_param(5000, start, params, data=y, burn.in = 1000))
```

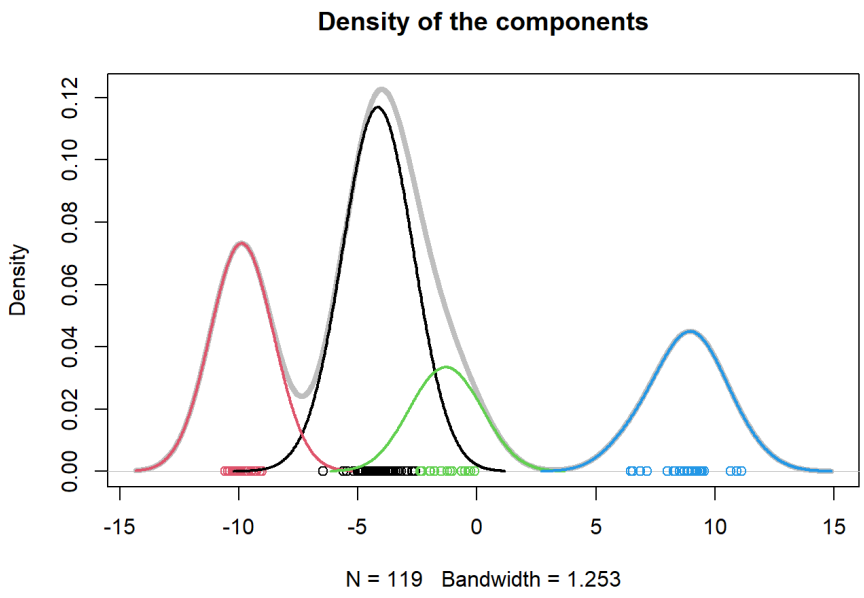
```
##    utente   sistema trascorso
##      8.50      0.02      9.36
```

As before for each component we can estimate μ_j, σ_j^2 and the number of points in each component.

```
C <- apply(result$C, 2, median)
mu <- apply(result$mu, 2, median)
sigma2 <- 1 / apply(result$A, 2, median)

N_j <- apply(as.matrix(1:k), 1, function(j) sum(C==j))
rbind(mu[N_j>0], sigma2[N_j>0], N_j[N_j>0])
```

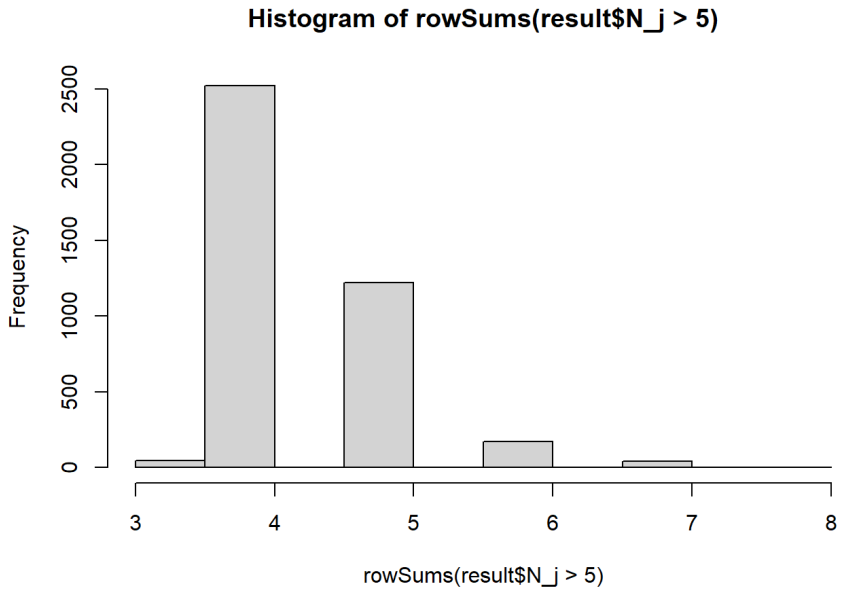
```
##           [,1]       [,2]      [,3]      [,4]
## [1,] -4.0977728 -9.8883792 -2.108508  8.603543
## [2,]  0.7981197  0.2536071  1.029154  1.370028
## [3,] 52.0000000 29.0000000 15.000000 23.000000
```

```
plot.clusters.densities(y, C, N_j/length(y))
```



As in previous exercise, a lot of clusters have a small number of elements. Thus, it makes more sense to not consider the clusters with a small number of elements.

```
hist(rowSums(result$N_j>5))
```



- Compare the results with the previous analysis in which the number of components was fixed a priori

The distance between the two estimates of μ is small

```
norm(sort(mu[N_j>0]) - sort(mu.2), "2")
```

```
## [1] 0.4739853
```

and also the mean silhoutte coefficient is very similar.

```
s <- mean(silhouette(y, C, k=sum(N_j>0)))
mean(s)
```

```
## [1] 0.7445383
```

```
mean(s) - mean(s.2)
```

```
## [1] 0.0006272241
```

This means that the result we obtain with the non-parametric approach is very similar to what we obtained assuming that $k = 4$.