732A73: Bayesian Learning

Computer Lab 3

Oriol Garrobé, Dávid Hrabovszki

06 May 2020

Question 1. Normal model, mixture of normal model with semi-conjugate prior

The data rainfall.dat consist of daily records, from the beginning of 1948 to the end of 1983, of precipitation (rain or snow in units of 1 100 inch, and records of zero precipitation are excluded) at Snoqualmie Falls, Washington.

(a)

First we analyze the date using a normal model. We assume that the daily precipitation $y_1, ..., y_n$ values are independent normally distributed:

$$y_1, ..., y_n | \mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$$

where μ, σ^2 are uknown. The prior distributions for these parameters are:

$$\mu \sim N(\mu_0, \tau_0^2)$$

$$\sigma^2 \sim Inv - \chi^2(\nu_0, \sigma_0^2)$$

(i)

Now we implement a Gibbs sampler that simulates from the joint posterior distribution:

$$p(\mu, \sigma^2 | y_1, ..., y_n)$$

.

The full conditional posteriors:

$$\mu | \sigma^2, x \sim \mathcal{N}(\mu_n, \tau_n^2)$$

$$\sigma^2 | \mu, x \sim Inv - \chi^2 \left(\nu_n, \frac{\nu_0 \sigma_0^2 + \sum_{i=1}^n (x_i - \mu)^2}{n + \nu_0} \right)$$

where

$$\mu_n = w\bar{x} + (1 - w)\mu_0$$

,

$$w = \frac{\frac{n}{\sigma^2}}{\frac{n}{\sigma^2} + \frac{1}{\tau_0^2}}$$

,

$$\frac{1}{\tau_n^2} = \frac{n}{\sigma^2} + \frac{1}{\tau_0^2}$$

We choose the following prior values: $\mu_0 = 30$, from looking at the data, because we had no prior knowledge in the topic. To express our uncertainty about this, we set the variance of the prior mean to a high value: $\tau_0^2 = 100$. Similarly, we choose $\sigma_0^2 = 1547.103$, which is the variance of the data sample. But then we pick a small prior for the degrees of freedom, because the prior is non-informative: $\nu_0 = 4$.

(ii)

In this part we analyze the daily precipitation using our Gibbs sampler.

Figure 1 shows the trajectory of the Markov chain for sampling the mean and the cumulative mean of the mean statistic as the number of iterations grows. We ran the sampler for 1000 iterations, but it seems clear that it converges after a very short time. We ignore the first 10 draws, as this seems to be a burn-in period, where the draws are not inside the distribution. The observed mean from the data is 32.283, while the mean of the draws is 32.285, so this indicates a successful Gibbs sampling. The Markov chain has a highly oscillating shape and the fact that the cumulative mean also converges to 32.28 means that the chain converges. The daily precipitation then has an average of around 32.28, but there is a possibility that it is between 31 and 33.5.

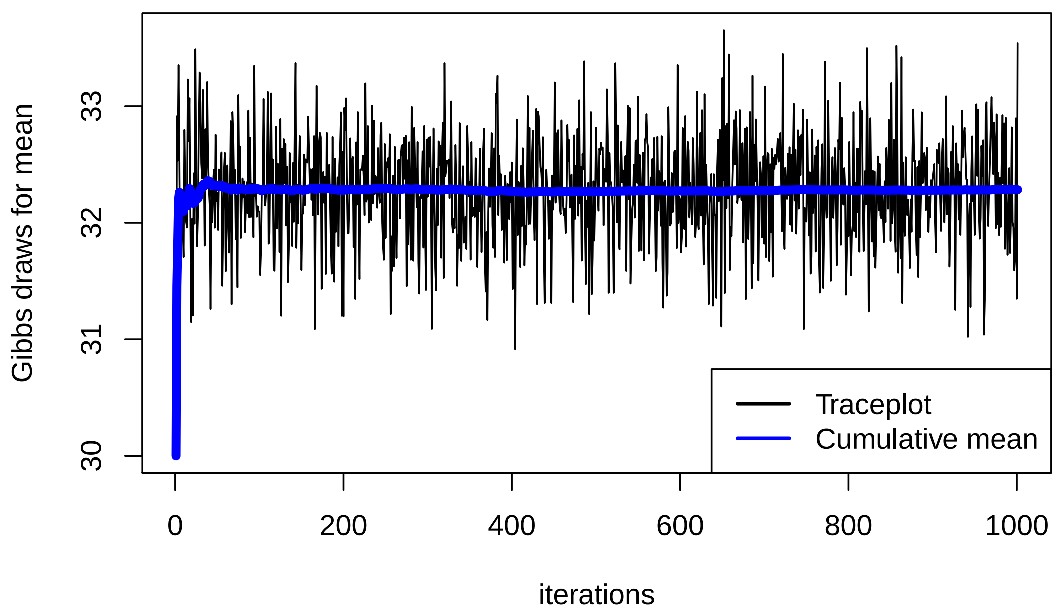


Figure 1: Markov chain of the sampled mean

Figure 2 shows the trajectory of the Markov chain for sampling the variance and the cumulative mean of the variance statistic as the number of iterations grows. We observe that the chain converges a bit more slowly than in the case of the mean, but it does so quite quickly nevertheless. Again, we ignore the burn in period (first 10 draws). The observed variance from the data is 1547.103, while the mean of the draws is 1546.152, so this indicates a successful Gibbs sampling. The Markov chain has a highly oscillating shape and the fact that the cumulative mean of the draws also converges to 1546.149 means that the chain converges. The daily precipitation then has a variance of around 1546.152, but there is a possibility that it is between 1475 and 1625.

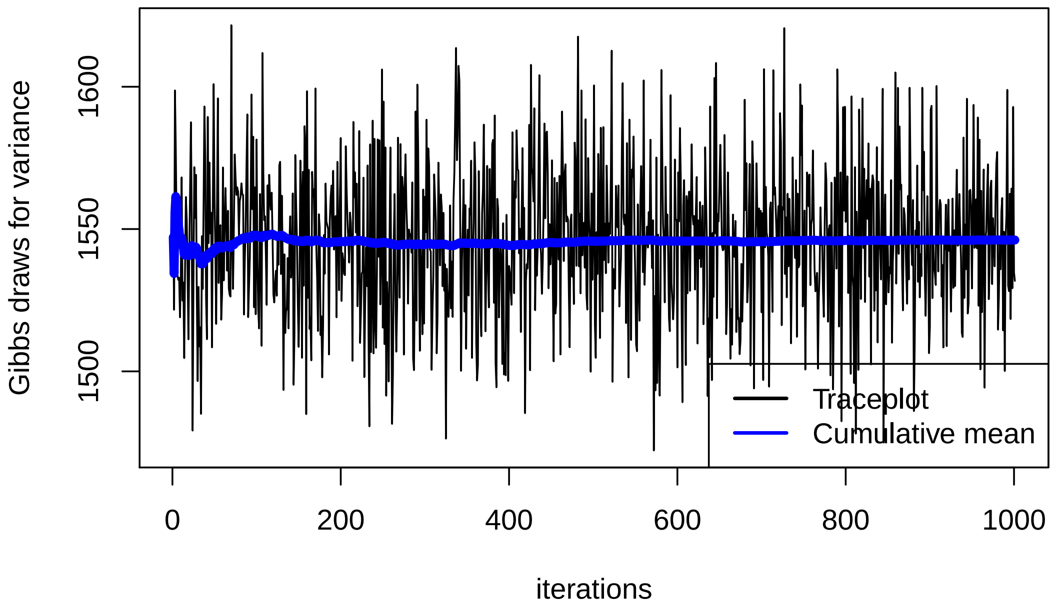


Figure 2: Markov chain of the sampled variance

Figures 3 and 4 show that the Gibbs samples for the mean and variance follow normal distributions, which is no surpise, since the priors were normal.

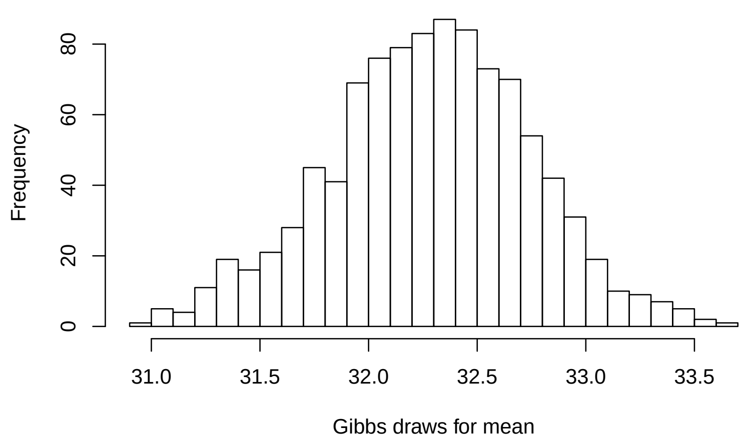


Figure 3: Histogram of the sampled mean

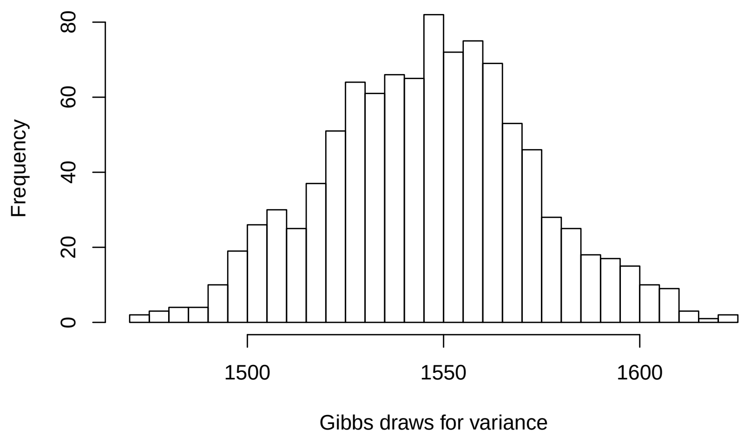


Figure 4: Histogram of the sampled variance

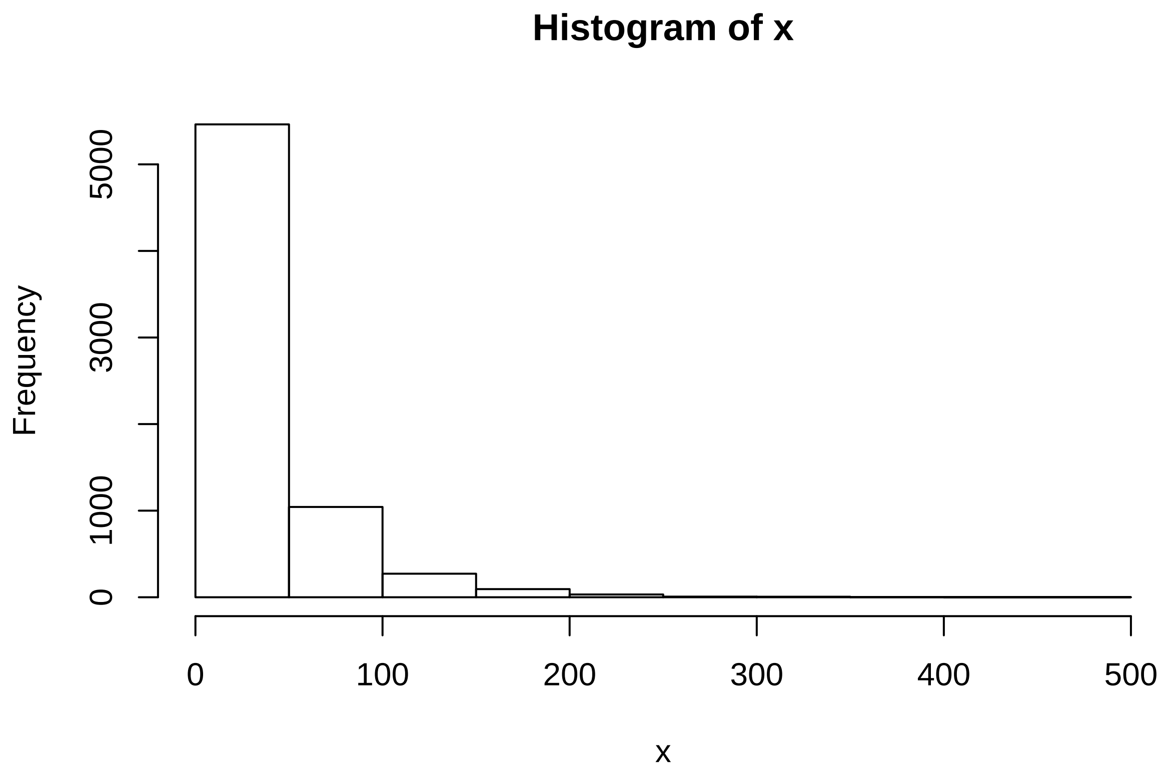
(b)

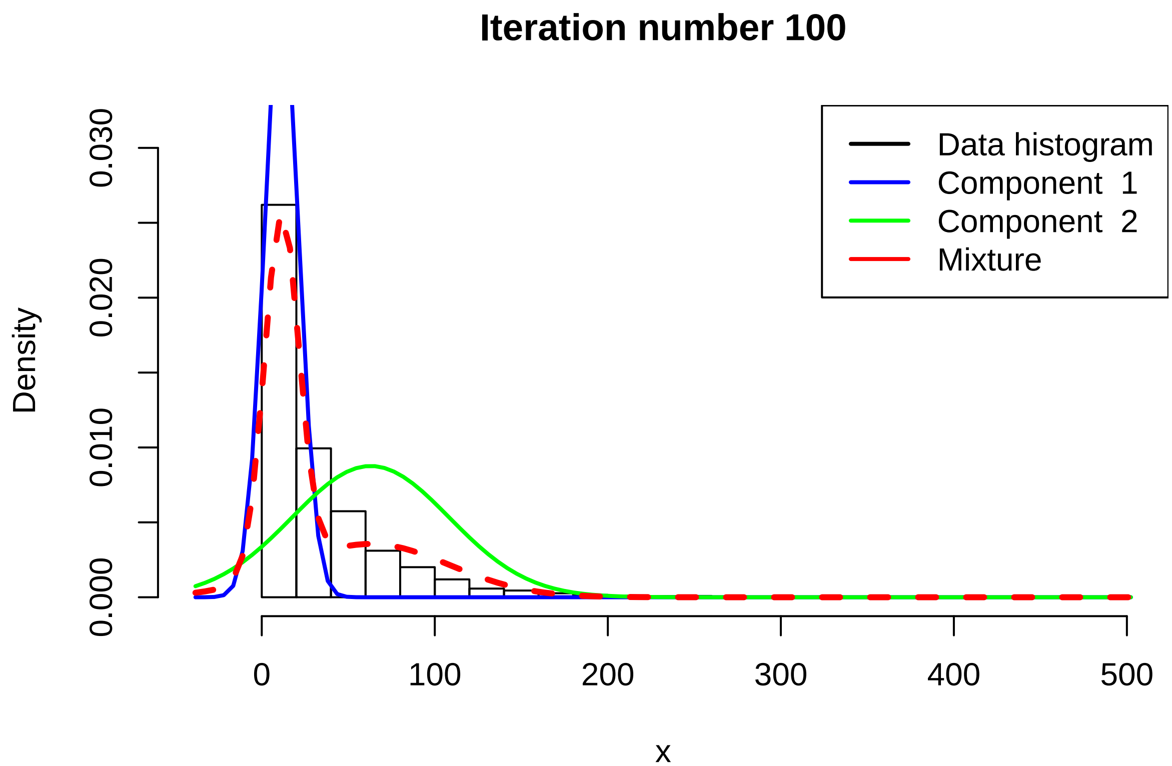
Here we assume that the daily precipitation $y_1, ..., y_n$ follows an iid. 2-component muxture of normals model:

$$p(y_i|\mu, \sigma^2, \pi) = \pi \mathcal{N}(y_i|\mu_1, \sigma_1^2) + (1-\pi)\mathcal{N}(y_i|\mu_2, \sigma_2^2)$$

where $\mu = (\mu_1, \mu_2)$ and $\sigma^2 = (\sigma_1^2, \sigma_2^2)$

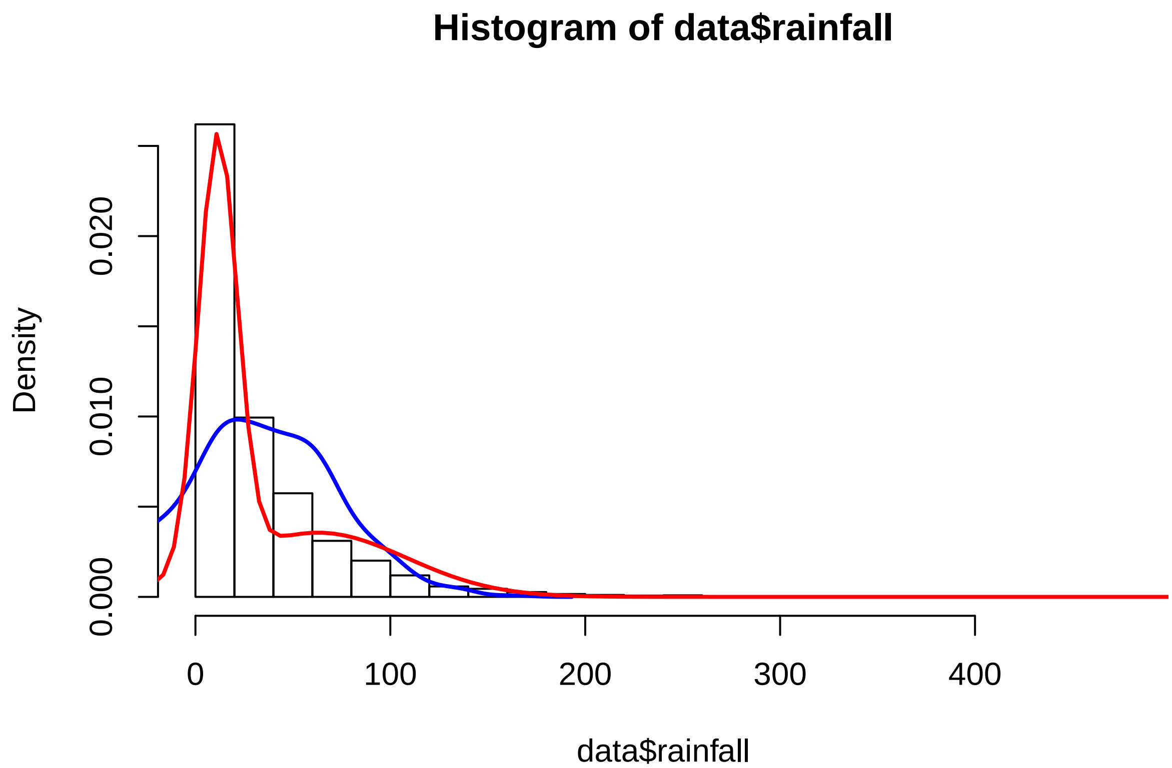
We use a slightly modified code from Mattias Villani from the file NormalMixtureGibbs.R to draw Gibbs samples from the mixture model. We set parameters the same way as in part a).


Histogram of x
Frequency
0
1000
3000
5000
0
100
200
300
400
500
x



Iteration number 100
Data histogram
Component 1
Component 2
Mixture
Density
0.000
0.010
0.020
0.030
0
100
200
300
400
500
x

(c)



Question 2. Metropolis Random Walk for Poisson regression

(a)

(b)

(c)

(d)

Appendix

```
knitr::opts_chunk$set(echo=FALSE, eval=TRUE)
# R version
RNGversion('3.5.1')

# libraries

#### 1.a
data = read.table('rainfall.dat', col.names = 'rainfall')

#### (i)

n = length(data$rainfall)
avg = mean(data$rainfall)

#prior setup
mu_0 = 30 #mu prior from prior knowledge, or looking at the data if there is no knowledge
tau2_0 = 100 #var(mu_0) high, because we have no prior knowledge
sigma2_0 = var(data$rainfall) # should use var(data$rainfall for best guess)
nu_0 = 4 #df, small, because we have no prior knowledge

# scaled-inverse chi-square simulator (from lab1 and from NormalMixtureGibbs.R)
rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

# Gibbs sampling
steps = 1000
gibbsDraws <- matrix(0,steps+1,2) #1st column: mu, 2nd column: sigma2
gibbsDraws[1,1] = mu_0
gibbsDraws[1,2] = sigma2_0

# Seed
set.seed(1234567890)
for (i in 1:steps) {
  #simulate mu from conditional posterior (normal)
  w = (n/gibbsDraws[i,2])/(n/gibbsDraws[i,2] + 1/tau2_0)
  mu_n = w * avg + (1-w) * mu_0
  tau2_n = 1 / (n/gibbsDraws[i,2] + 1/tau2_0)

  #update mu
  gibbsDraws[i+1,1] = rnorm(1, mu_n, sqrt(tau2_n))

  #simulate sigma2 from conditional posterior (inv-chisq)
  nu_n = nu_0 + n
  scale = (nu_0 * sigma2_0 + sum((data$rainfall - gibbsDraws[i+1,1])^2)) / (n + nu_0)

  #update sigma2
  gibbsDraws[i+1,2] = rScaledInvChi2(1, nu_n, scale)
}
#### (ii)
#Plot Markov chains and cumulative mean plots
```

```
cum_mean = numeric(steps + 1)
cum_var = numeric(steps + 1)
for (i in 1:(steps+1)) {
  cum_mean[i] = mean(gibbsDraws[1:i,1])
  cum_var[i] = mean(gibbsDraws[1:i,2])
}
plot(1:(steps+1), gibbsDraws[,1], type = 'l', xlab = 'iterations', ylab = 'Gibbs draws for mean')
legend(x='bottomright', legend=c('Traceplot', 'Cumulative mean'), col = c('black', 'blue'), lwd = 2)
lines(1:(steps+1), cum_mean, col = 'blue', lwd = 5)
plot(1:(steps+1), gibbsDraws[,2], type = 'l', xlab = 'iterations', ylab = 'Gibbs draws for variance')
legend(x='bottomright', legend=c('Traceplot', 'Cumulative mean'), col = c('black', 'blue'), lwd = 2)
lines(1:(steps+1), cum_var, col = 'blue', lwd = 5)

# result: mean of all posterior draws
burn_in_mu = 10
burn_in_sigma2 = 10
mu_gibbs = mean(gibbsDraws[burn_in_mu:length(gibbsDraws[,1]),1]) #32.28486
sigma2_gibbs = mean(gibbsDraws[burn_in_sigma2:length(gibbsDraws[,2]),2]) #1546.152

hist(gibbsDraws[burn_in_mu:length(gibbsDraws[,1]),1], breaks = 30, main = '', xlab = 'Gibbs draws for me

hist(gibbsDraws[burn_in_sigma2:length(gibbsDraws[,2]),2], breaks = 40, main = '', xlab = 'Gibbs draws fo

#### 1.b

#mixture model (normals)
# Code from NormalMixtureGibbs.R (by Mattias Villani)

# Estimating a simple mixture of normals
# Author: Mattias Villani, IDA, Linkoping University. http://mattiasvillani.com

##########    BEGIN USER INPUT #################
# Data options
x <- as.matrix(data$rainfall)

# Model options
nComp <- 2    # Number of mixture components

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(30,nComp) # Prior mean of mu
tau2Prior <- rep(100,nComp) # Prior std of mu: high, because we have no prior knowledge
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(4,nComp) # degrees of freedom for prior on sigma2: small, because we have no prior knowledge

# MCMC options
nIter <- 100 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green", "magenta", 'yellow')
sleepTime <- 0.1 # Adding sleep time between iterations for plotting
################   END USER INPUT ###############
```

```r
###### Defining a function that simulates from the
rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

####### Defining a function that simulates from a Dirichlet distribution
rDirichlet <- function(param){
  nCat <- length(param)
  piDraws <- matrix(NA,nCat,1)
  for (j in 1:nCat){
    piDraws[j] <- rgamma(1,param[j],1)
  }
  piDraws = piDraws/sum(piDraws) # Diving every column of piDraws by the sum of the elements in that co
  return(piDraws)
}

# Simple function that converts between two different representations of the mixture allocation
S2alloc <- function(S){
  n <- dim(S)[1]
  alloc <- rep(0,n)
  for (i in 1:n){
    alloc[i] <- which(S[i,] == 1)
  }
  return(alloc)
}
# Seed
set.seed(1234567890)

# Initial value for the MCMC
nObs <- length(x)
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp))) # nObs-by-nComp matrix with component all
mu <- quantile(x, probs = seq(0,1,length = nComp))
sigma2 <- rep(var(x),nComp)
probObsInComp <- rep(NA, nComp)

# Setting up the plot
xGrid <- seq(min(x)-1*apply(x,2,sd),max(x)+1*apply(x,2,sd),length = 100)
xGridMin <- min(xGrid)
xGridMax <- max(xGrid)
mixDensMean <- rep(0,length(xGrid))
effIterCount <- 0
ylim <- c(0,2*max(hist(x)$density))


for (k in 1:nIter){
  #message(paste('Iteration number:',k))
  alloc <- S2alloc(S) # Just a function that converts between different representations of the group al
  nAlloc <- colSums(S)
  #print(nAlloc)
  # Update components probabilities
  pi <- rDirichlet(alpha + nAlloc)

  # Update mu's
```

```r
  for (j in 1:nComp){
    precPrior <- 1/tau2Prior[j]
    precData <- nAlloc[j]/sigma2[j]
    precPost <- precPrior + precData
    wPrior <- precPrior/precPost
    muPost <- wPrior*muPrior + (1-wPrior)*mean(x[alloc == j])
    tau2Post <- 1/precPost
    mu[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }

  # Update sigma2's
  for (j in 1:nComp){
    sigma2[j] <- rScaledInvChi2(1, df = nu0[j] + nAlloc[j], scale = (nu0[j]*sigma2_0[j] + sum((x[alloc =
  }

  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- pi[j]*dnorm(x[i], mean = mu[j], sd = sqrt(sigma2[j]))
    }
    S[i,] <- t(rmultinom(1, size = 1 , prob = probObsInComp/sum(probObsInComp)))
  }

  # Printing the fitted density against data histogram
  if (k == nIter) {
    if (plotFit && (k%%1 ==0)){
    effIterCount <- effIterCount + 1
    hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = paste("Iteration number",k),
    mixDens <- rep(0,length(xGrid))
    components <- c()
    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens
      lines(xGrid, compDens, type = "l", lwd = 2, col = lineColors[j])
      components[j] <- paste("Component ",j)
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

    lines(xGrid, mixDens, type = "l", lty = 2, lwd = 3, col = 'red')
    legend("topright", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
    col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    #Sys.sleep(sleepTime)
  }
  }


}

# hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Final fitted density")
# lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")
# lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), type = "l", lwd = 2, col = "blue")
# legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"), col=
```

```
#########################    End of code from Mattias Villani    ########################################
#### c

#histogram of data
hist(data$rainfall, freq = F, breaks = 30)

#normal density from a)
norm_gibbs = rnorm(1000, mean = mu_gibbs, sd = sqrt(sigma2_gibbs))
lines(density(norm_gibbs), col = 'blue', lwd = 2)

#mixture of normals from b)
lines(xGrid, mixDensMean, type = "l", lwd = 2, col = "red")
```