Lab3

Naveen_Gabriel

19 may 2019

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

1. Normal model.

Assuming the daily precipitaion follows normal distribution whre both μ and σ^2 are unknown.

$$y_1, y_2 ... y_n \sim \mathcal{N}(\mu, \sigma^2)$$

Assuming our semi-conjugate priors μ and σ follows below distribution :

$$\mu \sim \mathcal{N}(\mu_0, tau_0^2)$$
$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

So our full conditional posterior is :

$$\mu|\sigma^2, x \sim \mathcal{N}(\mu_n, tau_n^2)$$
$$\sigma^2|\mu, x \sim Inv - \chi^2(v_n, \frac{v_o\sigma^2 + \sum_{i=0}^{n}(x_i - \mu)^2}{n + v_0})$$

Where v_n and tau_n^2 values are

$$\tau_n^2 = \frac{\sigma^2\tau_0^2}{n\tau_0^2 + \sigma^2}$$
$$\mu_n = w\bar{x} + (1 - w)\mu_0$$

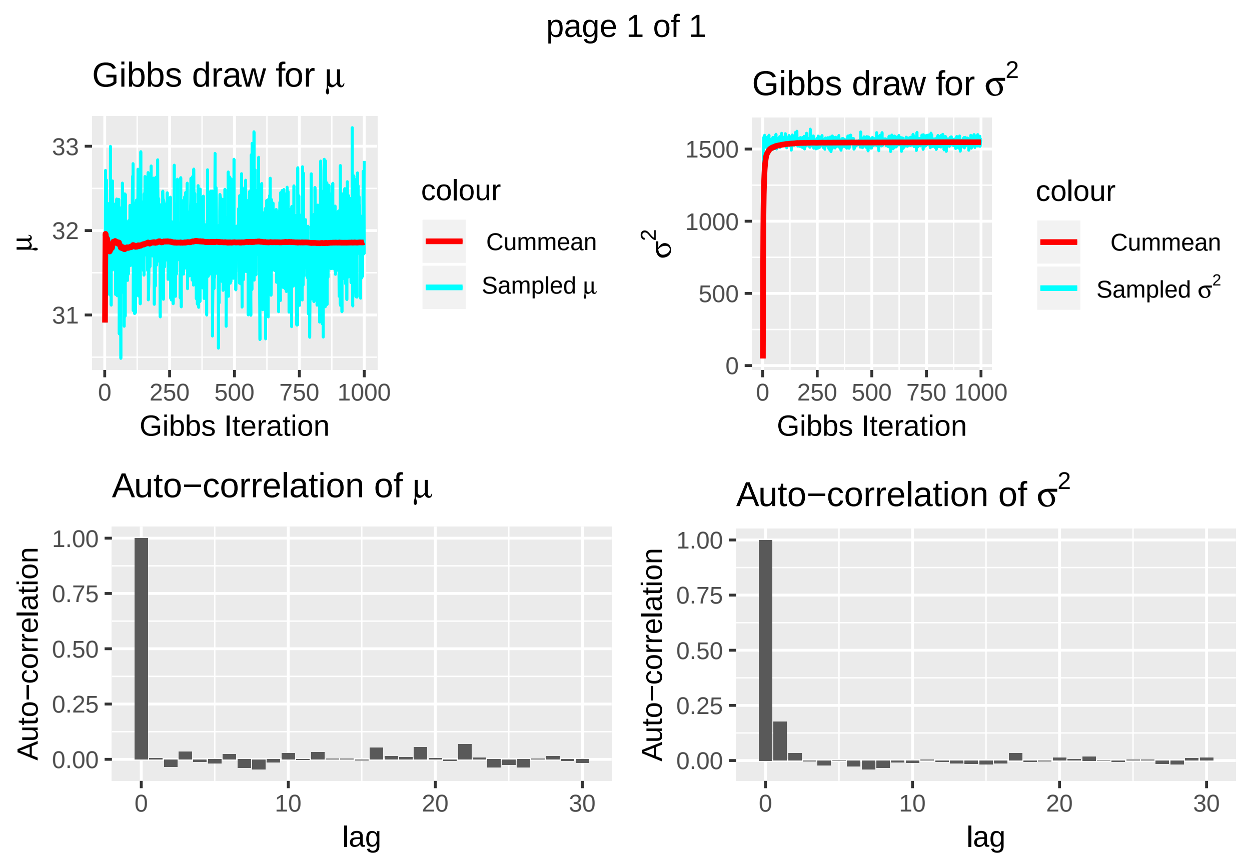
Where w is :

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

```r
rainfall <- read.table("rainfall.txt")
colnames(rainfall) <- "Precipitation"

n <- nrow(rainfall)

#Initial parameters
mu0 <- 30
sg0_sq <- 50
tau0_sq <- 1
nDraws = 1000  #number of draws for parameters
v0 <- 1500


gibbs_bayes <- function(x,mu,sg_sq,tau0_sq,v0)
```

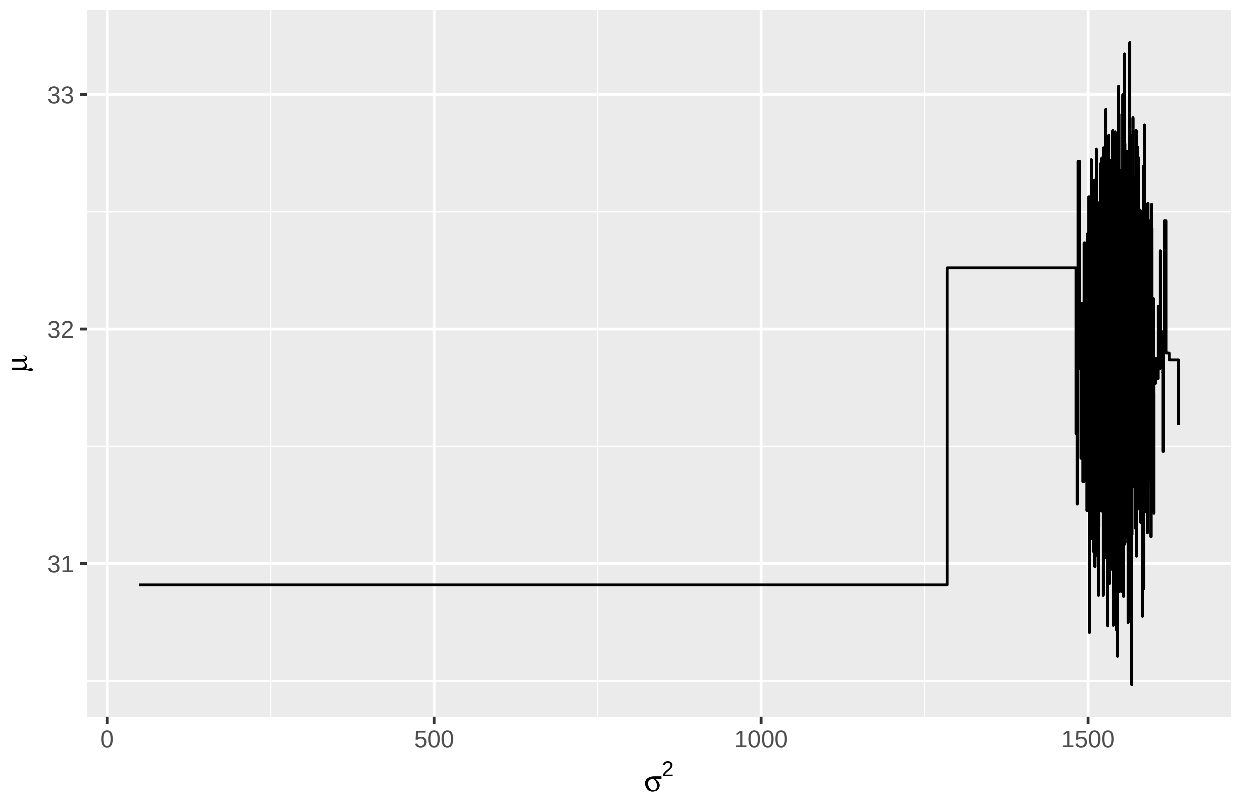
```r
{ mn <- mean(x)
  n <- length(x)
  vn <- v0 +n
  gibbsDraws <- matrix(0,nDraws,2)
  gibbsDraws[1,] <- c(mu,sg_sq)
  for (i in 2:nDraws) {

    numertr <- (n/sg_sq)
    denomtr <- (n/sg_sq) + (1/tau0_sq)
    w <- (numertr)/(denomtr)
    mu_n <- w*mn + (1-w)*mu0

    taun_sq <-  (sg_sq*tau0_sq)/((n*tau0_sq)+ sg_sq)
    mu <- rnorm(1,mu_n,sqrt(taun_sq))

    scn_arg <- ((v0*sg_sq)+sum((x-mu)^2))/vn
    sg_sq <- invchisq(vn,scn_arg)
    gibbsDraws[i,] <- c(mu,sg_sq)

  }

  return(gibbsDraws)

}

mu <- rnorm(1,mu0,sqrt(tau0_sq))
sg_sq <- invchisq(v0,sg0_sq)

data <- gibbs_bayes(rainfall$Precipitation,mu,sg_sq,tau0_sq,v0)
data <- as.data.frame(data)
colnames(data) <- c("mu","sig")
data$cummean_mu <- cummean(data$mu)
data$cummean_sig <- cummean(data$sig)

#calculating autocorrelation

mu_ac <- acf(data$mu,plot=FALSE,type="correlation")
sig_ac <- acf(data$sig,plot=FALSE,type="correlation")
autocor_data <- data.frame(mu_ac$acf,sig_ac$acf,lag=mu_ac$lag)
colnames(autocor_data) <- c("mu_ac","sig_ac","lag")

plot_gib <- c()
plot_gib[[1]] <- ggplot(data,aes(x=1:nDraws,y=mu,color="mu")) + geom_line() + xlab("Gibbs Iteration") +
                                          ylab(expression(mu)) +
                                          geom_line(aes(x=1:nDraws,y=cummean_mu,color="cummean_mu"),siz
                                          ggtitle(expression(paste("Gibbs draw for ",mu))) +
              scale_color_manual(labels = c("Cummean",expression(paste("Sampled ",mu))), values = c("re

plot_gib[[2]] <- ggplot(data=autocor_data, mapping=aes(x=lag, y=mu_ac)) +
                 geom_bar(stat = "identity", position = "identity") +
                 ggtitle(expression(paste("Auto-correlation of ",mu))) + ylab("Auto-correlation")

plot_gib[[3]] <- ggplot(data,aes(x=1:nDraws,y=sig,color="sig")) + geom_line() + xlab("Gibbs Iteration")
```

```
                                          ylab(expression(sigma^2)) +
                                          geom_line(aes(x=1:nDraws,y=cummean_sig,color="cummean_mu"),si
                                          ggtitle(expression(paste("Gibbs draw for ",sigma^2))) +
        scale_color_manual(labels = c("Cummean", expression(paste("Sampled ", sigma^2))), values = c("r

plot_gib[[4]] <- ggplot(data=autocor_data, mapping=aes(x=lag, y=sig_ac)) +
               geom_bar(stat = "identity", position = "identity") +
               ggtitle(expression(paste("Auto-correlation of ", sigma^2))) +
               ylab("Auto-correlation")

marrangeGrob(plot_gib,nrow=2, ncol=2)
```



```
ggplot(data,aes(x=sig,y=mu)) + geom_step() + xlab(expression(sigma^2)) + ylab(expression(mu)) + ggtitle
```

Convergence of gibbs sampler:Gaussian space



From the above plots, it is clear that using gibbs sampler converges to stable mu and sigma value

2 Mixture Model

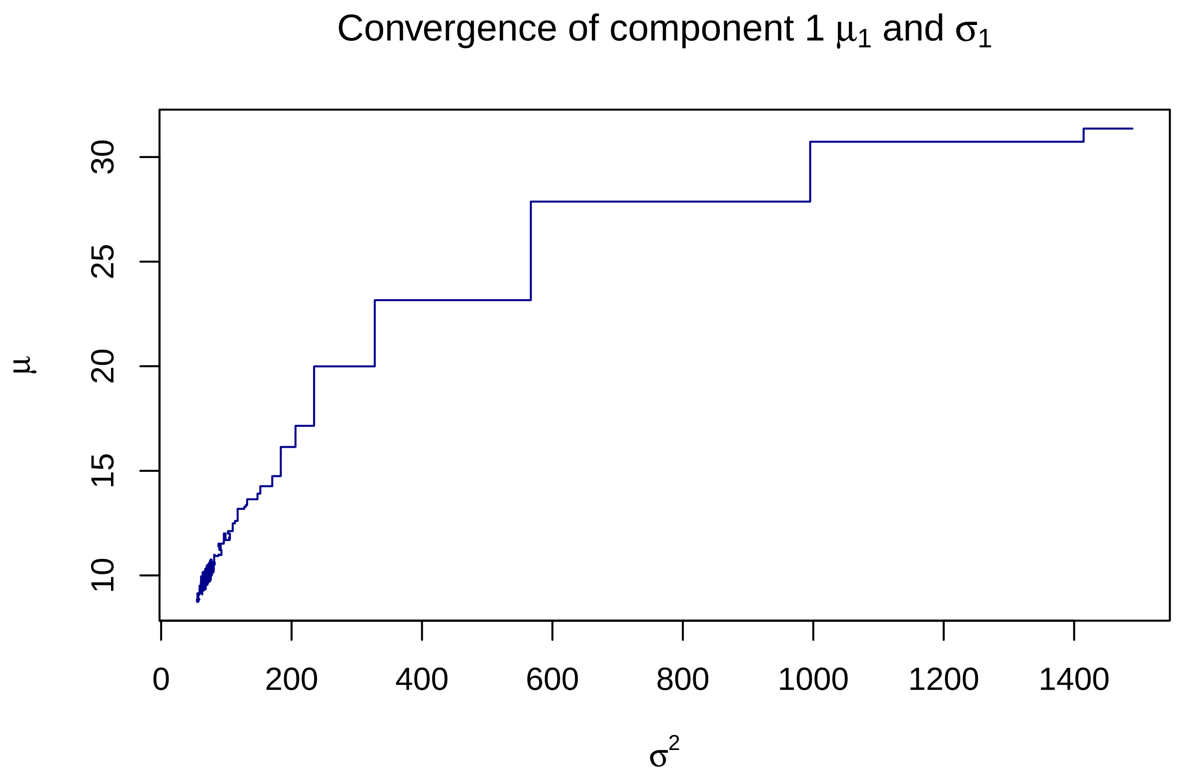
```
#Code credits :  Estimating a simple mixture of normals Author: Mattias Villani
#this uses semi conjugate prior
# Data options
set.seed(12345)
x <- as.matrix(rainfall$Precipitation)

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

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(0,nComp) # Prior mean of mu
tau2Prior <- rep(10,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(4,nComp) # degrees of freedom for prior on sigma2

# MCMC options
nIter <- 1000 # Number of Gibbs sampling draws

# Plotting options
plotFit <- TRUE
```

```r
lineColors <- c("blue", "green", "magenta", 'yellow')
################   END USER INPUT ###############

###### 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)
  }
# Diving every column of piDraws by the sum of the elements in that column.
  piDraws = piDraws/sum(piDraws)
  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){
    # from all components it return which position has 1
    alloc[i] <- which(S[i,] == 1)
  }
  return(alloc)
}

# Initial value for the MCMC
nObs <- length(x)
#S is a matrix which says from which component the
#particular observation is sampled'
# nObs-by-nComp matrix with component allocations.
S <- t(rmultinom(nObs, size = 1 , prob = rep(1/nComp,nComp)))
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(density(x)$y))
#ylim = c(0,2*max(hist(x)$density))
```

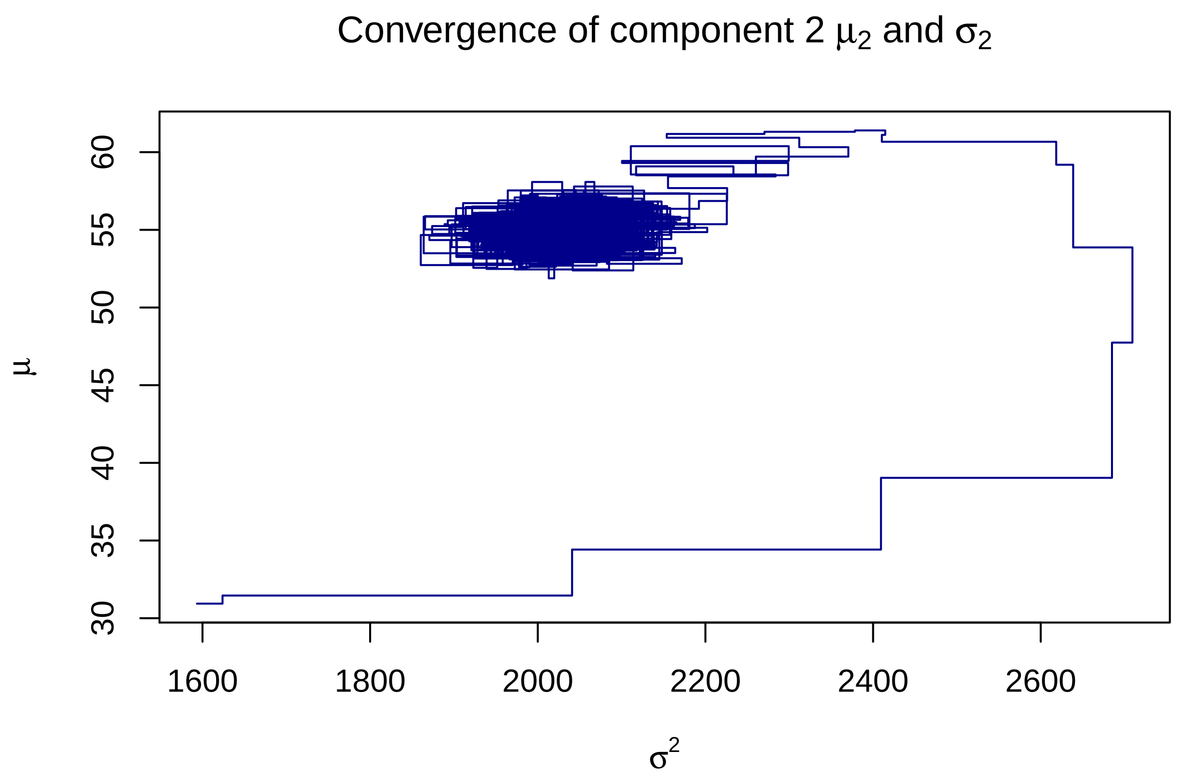
```r
par_mat = matrix(NA, nrow = nIter, ncol = nComp*2)
for (k in 1:nIter){
  alloc <- S2alloc(S) # Just a function that converts between different
                      #representations of the group allocations
  nAlloc <- colSums(S)
  # Update components probabilities
  pi <- rDirichlet(alpha + nAlloc)

  # Update mu's
  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))
  }
   par_mat[k,1:nComp] = mu

  # 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 == j] - mu[j])^2))/(nu0[j] + nAlloc[j]))

  }
  par_mat[k, -c(1:nComp)] = sigma2

  # 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 (plotFit && (k%%1 ==0)){
    effIterCount <- effIterCount + 1

    mixDens <- rep(0,length(xGrid))

    for (j in 1:nComp){
      compDens <- dnorm(xGrid,mu[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + pi[j]*compDens
    }
    mixDensMean <- ((effIterCount-1)*mixDensMean + mixDens)/effIterCount

  }

}
```

Convergence of sampler

```r
plot(x = par_mat[,3], y = par_mat[,1], type = 's',
     main=expression(paste("Convergence of component 1 ",mu[1]," and ", sigma[1])),
     col = "DarkBlue", xlab=expression(paste("",sigma^2)),
     ylab=expression(paste("",mu)))
```

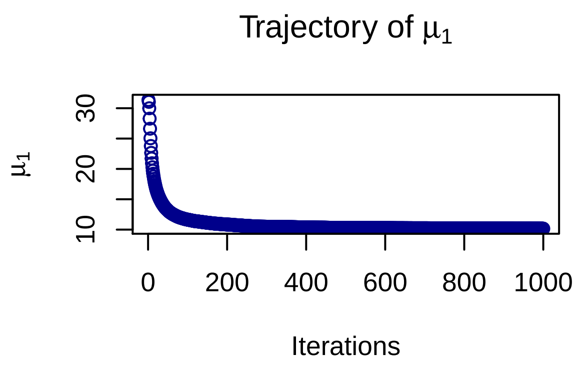


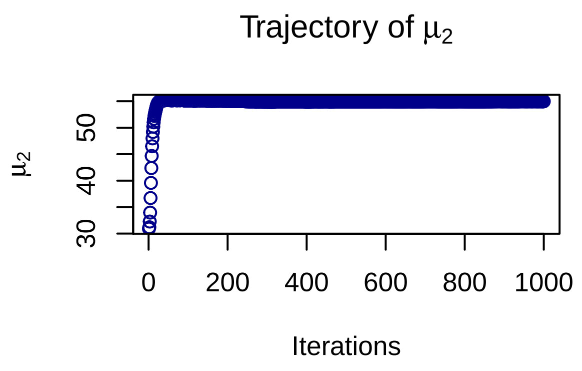
```r
plot(x = par_mat[,4], y = par_mat[,2], type = 's',
     main=expression(paste("Convergence of component 2 ",mu[2]," and ", sigma[2])),
     col = "DarkBlue",xlab=expression(paste("",sigma^2)),
     ylab=expression(paste("",mu)))
```

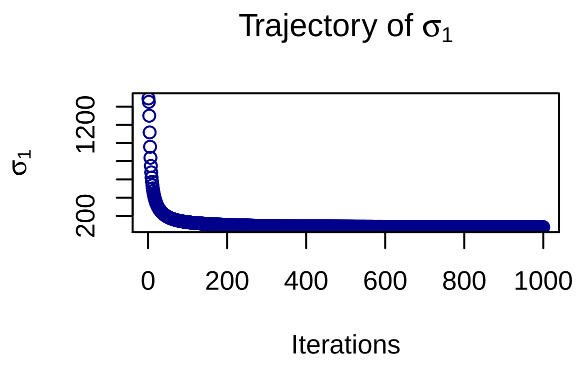


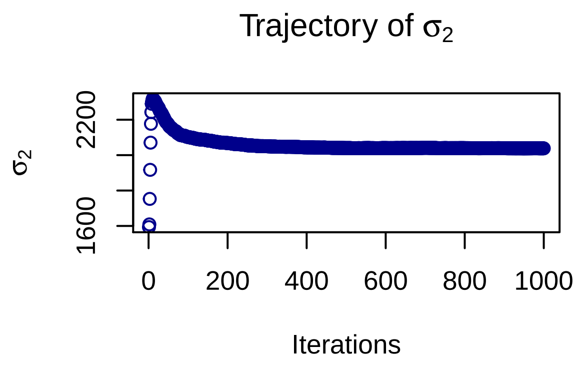
Parameter Trajectories

```
par_mat_cumsum = apply(par_mat, 2, cumsum)
par_mat_cumsum = apply(par_mat_cumsum,2,"/",seq(1:nIter))

par(mfrow =c(2,2))
plot(x = seq(1:nIter) , y = par_mat_cumsum[,1], col = "DarkBlue" ,
     main = expression(paste("Trajectory of ",mu[1])), xlab = "Iterations",
     ylab = expression(paste(mu[1])))

plot(x = seq(1:nIter) , y = par_mat_cumsum[,2], col = "DarkBlue" ,
     main = expression(paste("Trajectory of ",mu[2])), xlab = "Iterations",
     ylab = expression(paste(mu[2])))

plot(x = seq(1:nIter) , y = par_mat_cumsum[,3], col = "DarkBlue" ,
     main = expression(paste("Trajectory of ",sigma[1])), xlab = "Iterations",
     ylab = expression(paste(sigma[1])))

plot(x = seq(1:nIter) , y = par_mat_cumsum[,4], col = "DarkBlue" ,
    main = expression(paste("Trajectory of ",sigma[2])), xlab = "Iterations",
    ylab = expression(paste(sigma[2])))
```







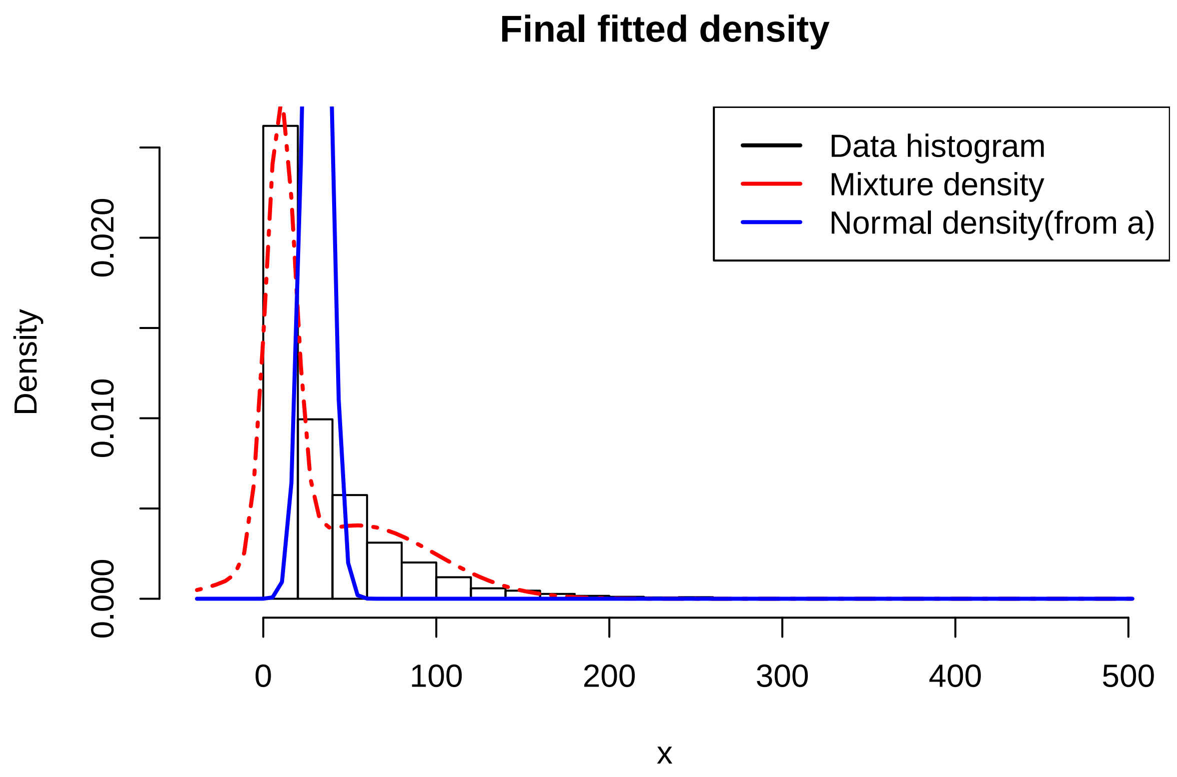


When we consider 2 component mixture of normel model for iid distribution of precipitation, the convergance of sampler and component parameters are visualized using the plots above.

Graphical Comparison

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

```
hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax),
     main = "Final fitted density")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 6, col = "red")
lines(xGrid, dnorm(xGrid, mean = Mode(data$mu), sd = sqrt(Mode(data$sig))),
      type = "l", lwd = 2, col = "blue")
legend("topright", box.lty = 1,
       legend = c("Data histogram","Mixture density","Normal density(from a)"), col=c("black","red","blu
```



For plotting the normal density from part a, we have considered the mode of sampled values from mu and sigma and using that the normal distribution was plotted for the range of values.