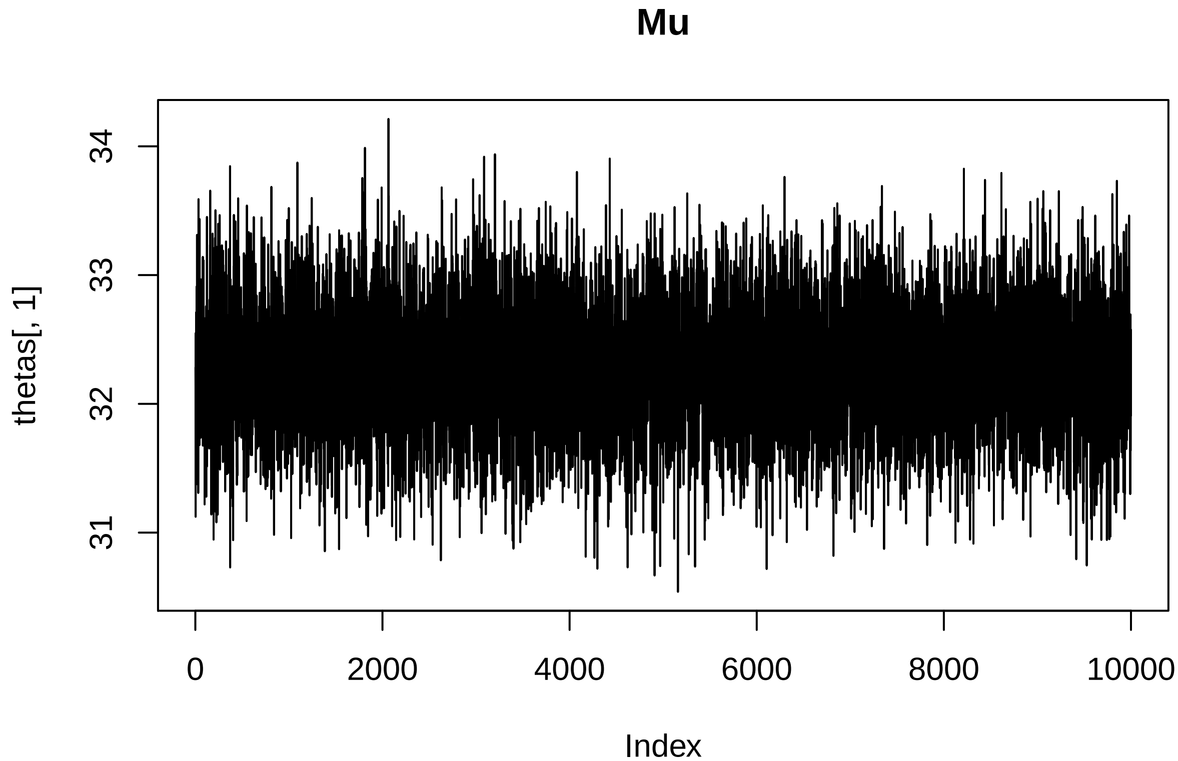
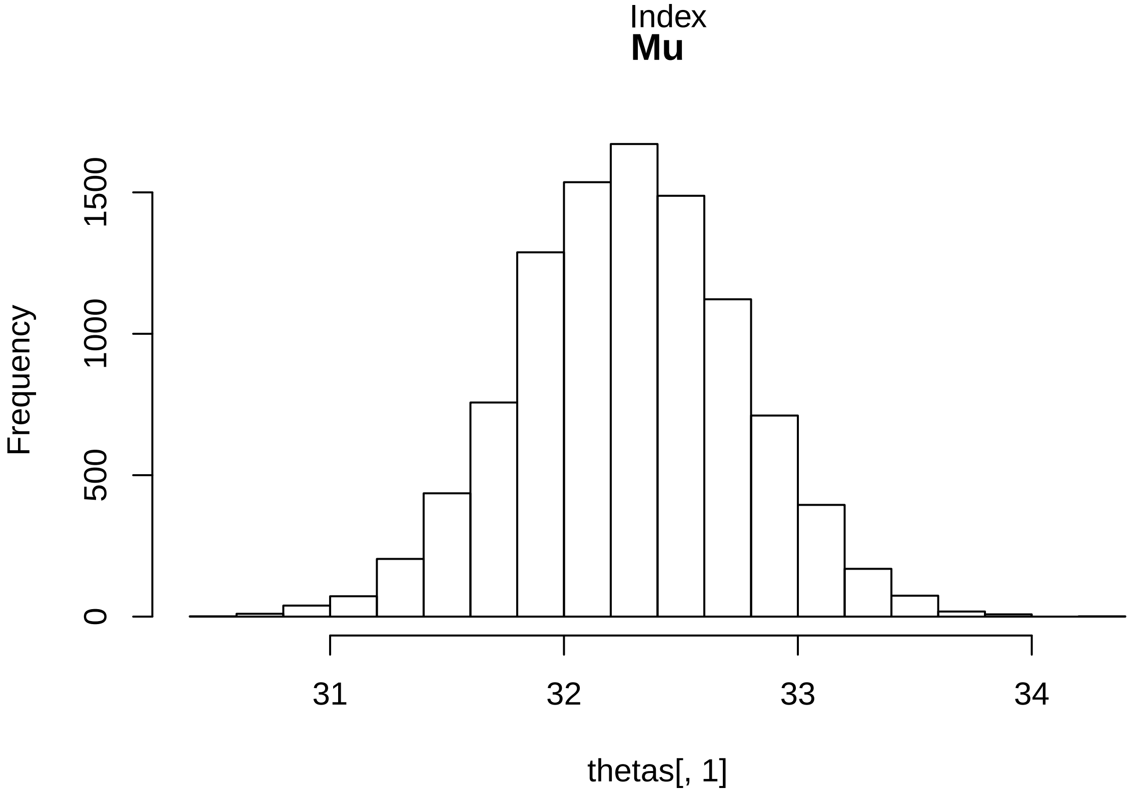
Lab_3_report

Axel Holmberg (axeho681), Wilhelm Hansson (wilha431)

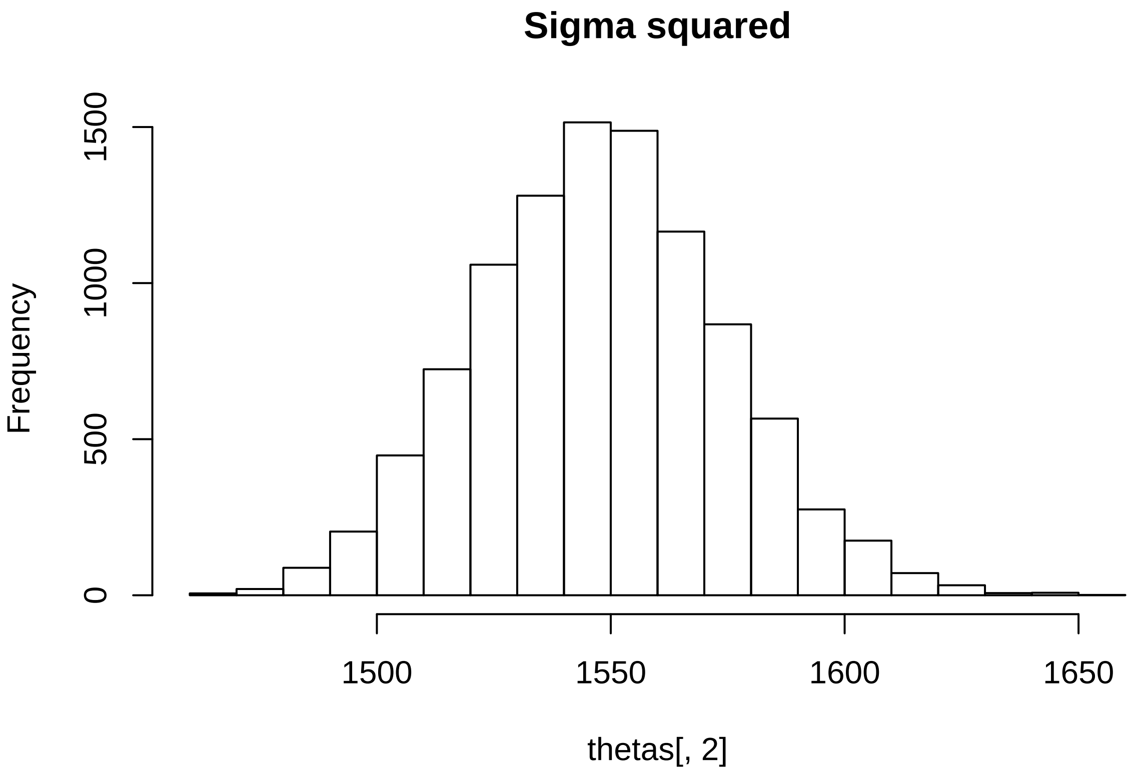
1

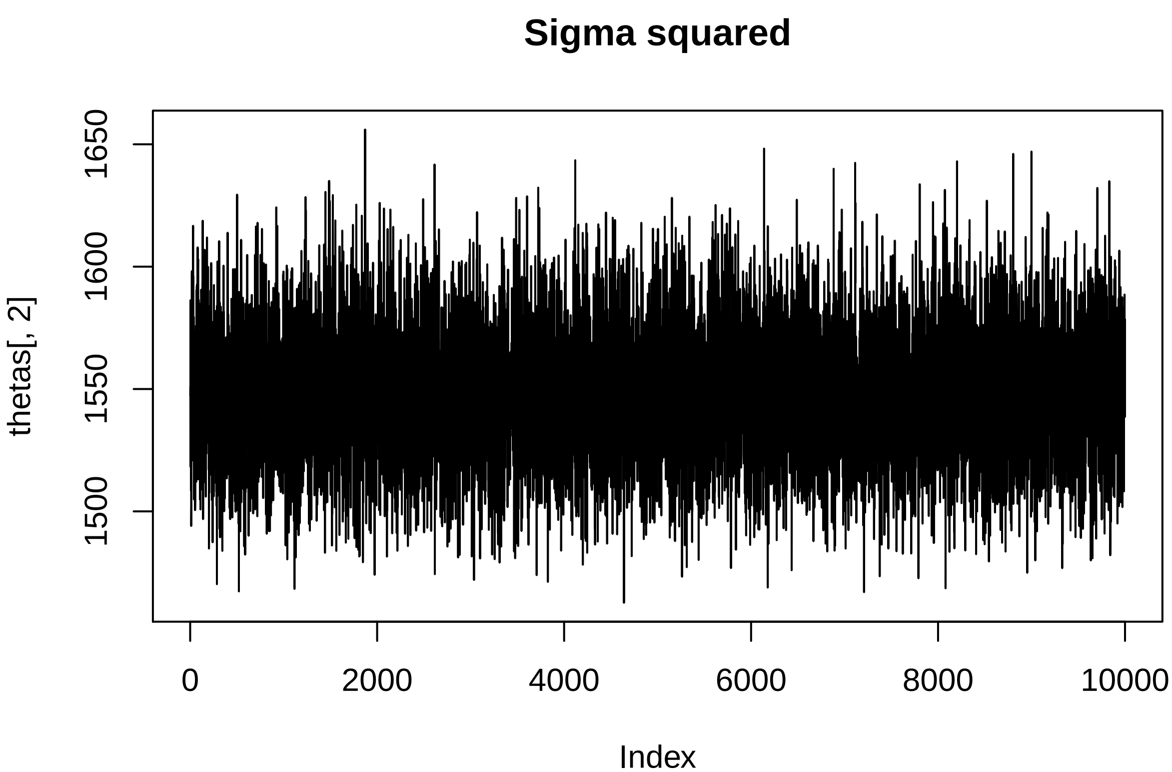
a)





As one can see above the values from the Gibbs model varies as it simulates μ for the joint posterior.





As one can see above the values from the Gibbs model varies as it simulates σ^2 for the joint posterior.

b)

The plots above shows iteration 1, 3, 5 and 1000 of the Gibbs sampling data augmentation algorithm. As one can see it starts at the same place, but as it iterates it splits into two normal distributions, each describing a

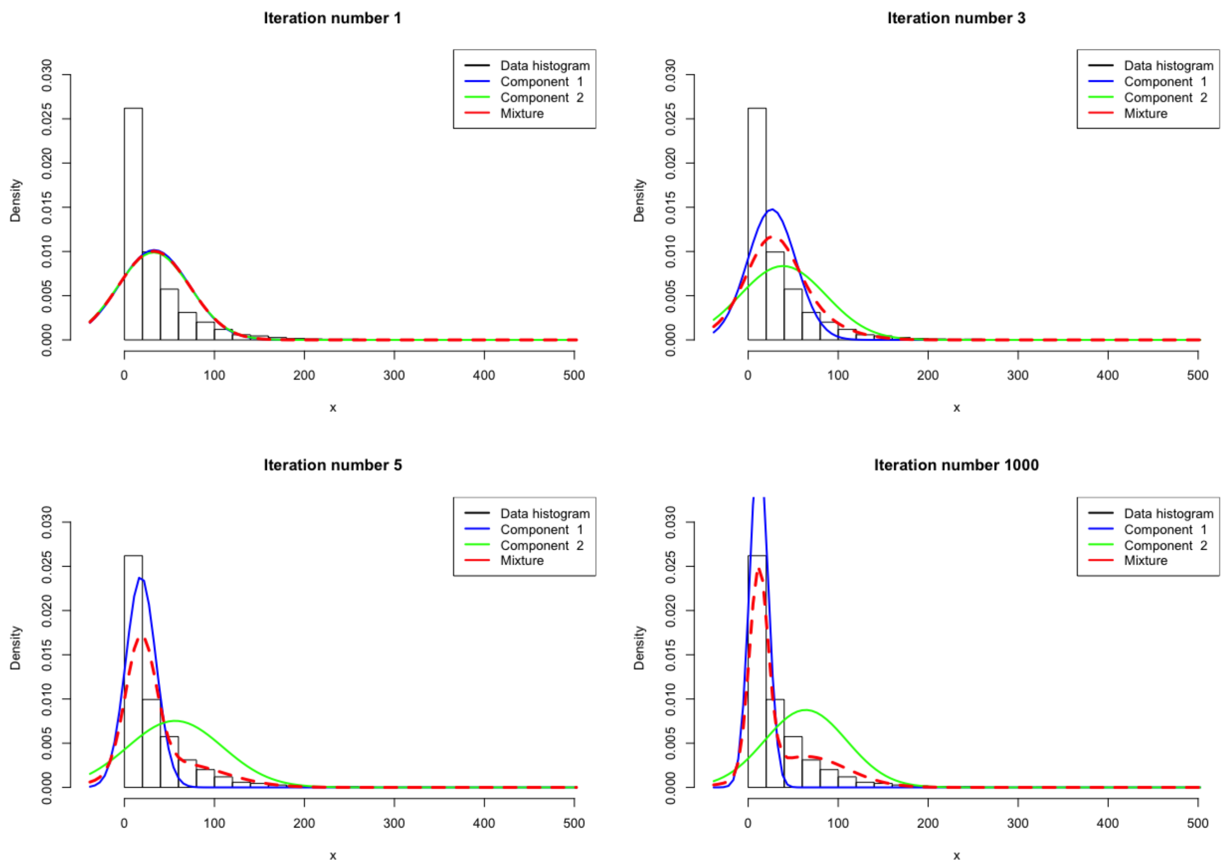


Figure 1: Iterations

part of the data well. The combined value of this can be seen in the dashed red line showing the Mixture normal model.

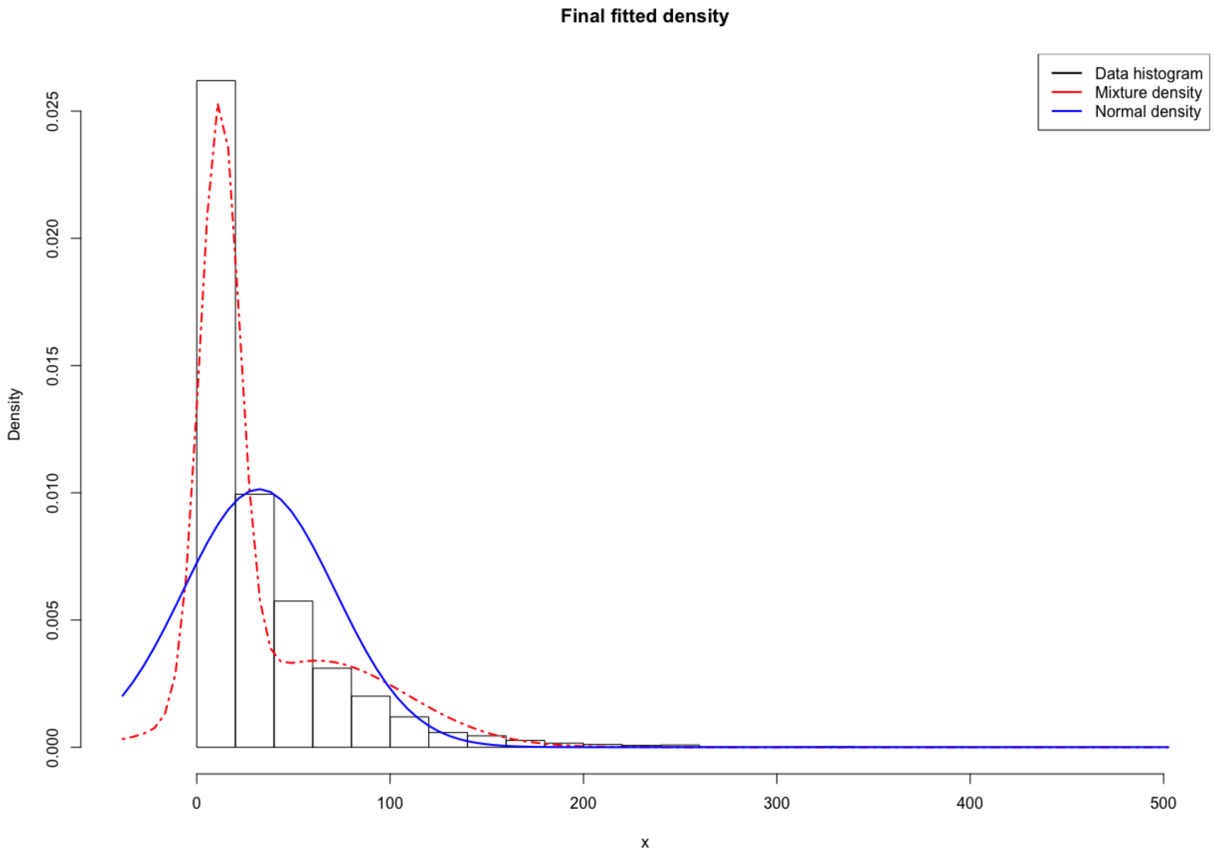


Figure 2: Final Fitted

The plot above shows the final fitted version of the mixture model as well as a normal distribution based on the mean and variance of the original data.

c)

The plot above shows the final fitted version of the mixture model as well as a normal distribution created from values from the Gibbs sampler from 1 a).

One can also see that the normal distribution based on the mean and variance of the original data is almost the exact same as the normal distribution created from values from the Gibbs sampler from 1 a).

2

a)

```
##
## Call:  glm(formula = nBids ~ . - Const, family = poisson, data = data)
##
## Coefficients:
## (Intercept)  PowerSeller     VerifyID       Sealed      Minblem      MajBlem
##     1.07244     -0.02054     -0.39452      0.44384     -0.05220     -0.22087
```

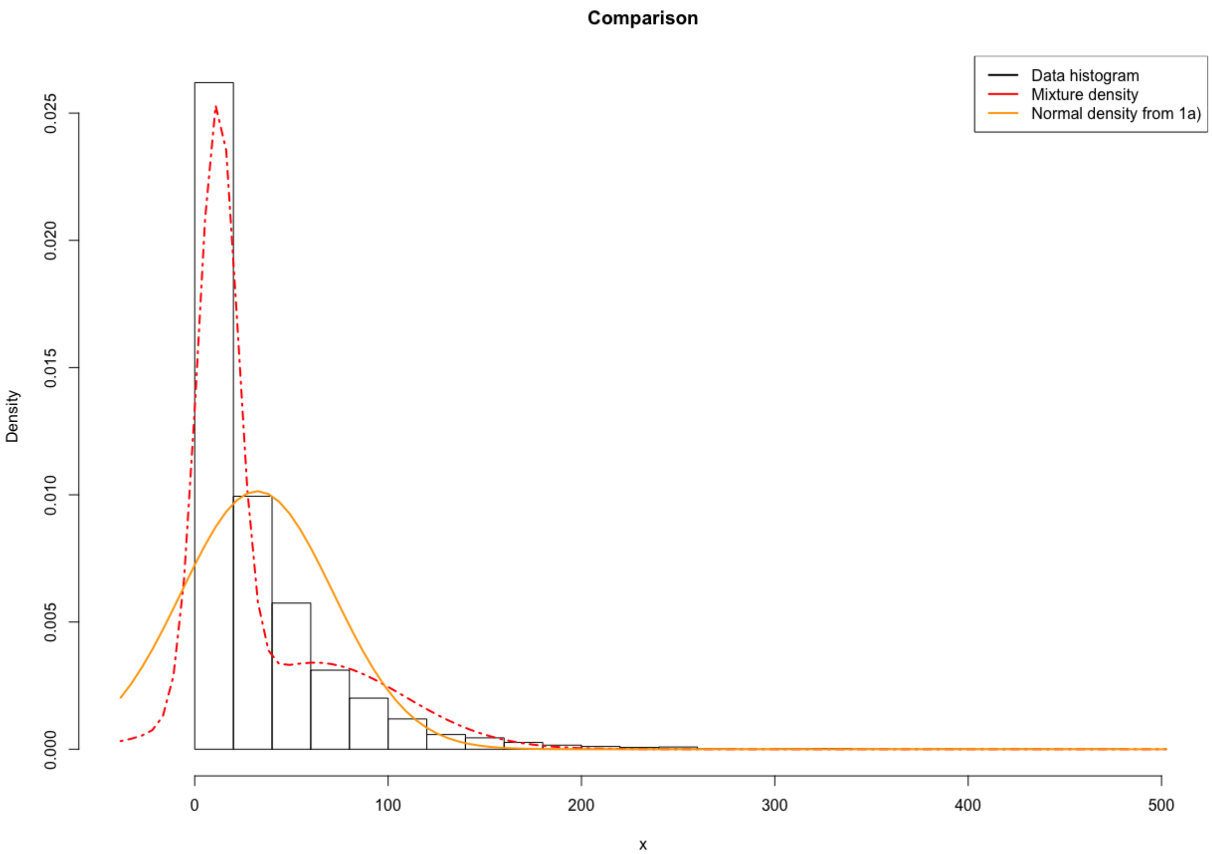


Figure 3: Comparison

```
##     LargNeg      LogBook  MinBidShare
##     0.07067     -0.12068     -1.89410
##
## Degrees of Freedom: 999 Total (i.e. Null);  991 Residual
## Null Deviance:       2151
## Residual Deviance: 867.5     AIC: 3610
```

The interesting part to note in the model above are the coefficients for each variable as these will be compared to in b) and c). The ones that are most significant are MinBidShare, Sealed and VerifyID.

b)

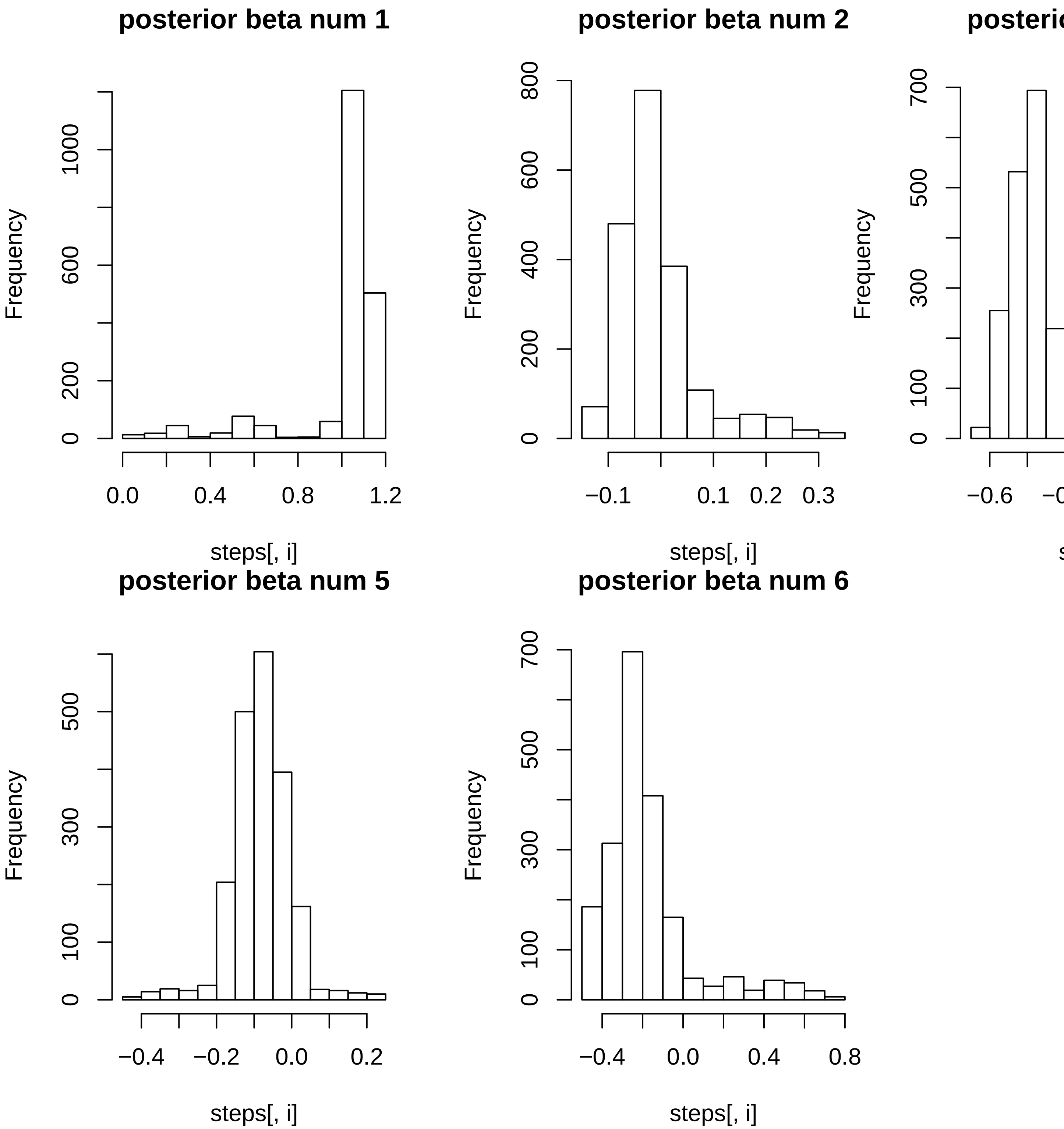
$J_y^{-1}(\tilde{\beta})$ is:

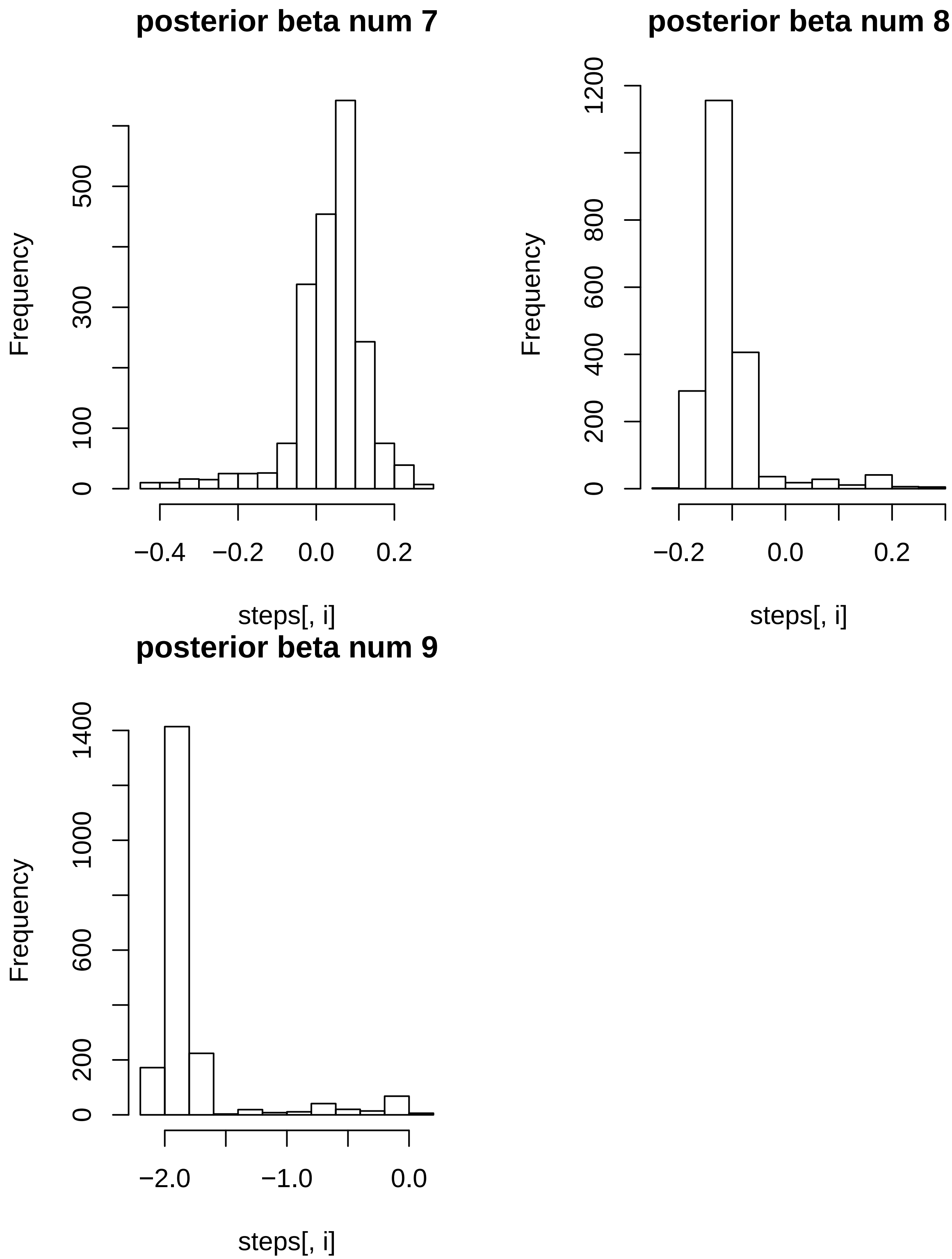
```
##               [,1]          [,2]          [,3]          [,4]          [,5]
##  [1,]  0.0009454637 -7.138975e-04 -2.741522e-04 -2.709019e-04 -4.454555e-04
##  [2,] -0.0007138975  1.353076e-03  4.025159e-05 -2.948981e-04  1.142965e-04
##  [3,] -0.0002741522  4.025159e-05  8.515389e-03 -7.824884e-04 -1.013608e-04
##  [4,] -0.0002709019 -2.948981e-04 -7.824884e-04  2.557755e-03  3.577152e-04
##  [5,] -0.0004454555  1.142965e-04 -1.013608e-04  3.577152e-04  3.624656e-03
##  [6,] -0.0002772239 -2.082672e-04  2.282520e-04  4.532315e-04  3.492348e-04
##  [7,] -0.0005128360  2.801775e-04  3.313565e-04  3.376477e-04  5.843965e-05
##  [8,]  0.0000643685  1.181858e-04 -3.191843e-04 -1.311039e-04  5.854069e-05
##  [9,]  0.0011099386 -5.685709e-04 -4.292772e-04 -5.759436e-05 -6.437103e-05
##               [,6]          [,7]          [,8]          [,9]
##  [1,] -2.772239e-04 -5.128360e-04  6.436850e-05  1.109939e-03
##  [2,] -2.082672e-04  2.801775e-04  1.181858e-04 -5.685709e-04
##  [3,]  2.282520e-04  3.313565e-04 -3.191843e-04 -4.292772e-04
##  [4,]  4.532315e-04  3.376477e-04 -1.311039e-04 -5.759436e-05
##  [5,]  3.492348e-04  5.843965e-05  5.854069e-05 -6.437103e-05
##  [6,]  8.365167e-03  4.048650e-04 -8.975867e-05  2.622259e-04
##  [7,]  4.048650e-04  3.175041e-03 -2.541761e-04 -1.063185e-04
##  [8,] -8.975867e-05 -2.541761e-04  8.384722e-04  1.037431e-03
##  [9,]  2.622259e-04 -1.063185e-04  1.037431e-03  5.054768e-03
```

$\tilde{\beta}$ is:

```
## [1]  1.06984022 -0.02051279 -0.39301376  0.44356933 -0.05248257 -0.22125193
## [7]  0.07070676 -0.12022017 -1.89199115
```

c)


posterior beta num 1
Frequency
0
200
600
1000
0.0
0.4
0.8
1.2
steps[, i]
posterior beta num 2
Frequency
0
200
400
600
800
−0.1
0.1
0.2
0.3
steps[, i]
Frequency
0
100
300
500
700
−0.6
posterior beta num 5
Frequency
0
100
300
500
−0.4
−0.2
0.0
0.2
steps[, i]
posterior beta num 6
Frequency
0
100
300
500
700
−0.4
0.0
0.4
0.8
steps[, i]


posterior beta num 7
Frequency
0
100
300
500
−0.4
−0.2
0.0
0.2
steps[, i]
posterior beta num 8
Frequency
0
200
400
600
800
1200
−0.2
0.0
0.2
steps[, i]
posterior beta num 9
Frequency
0
200
600
1000
1400
−2.0
−1.0
0.0
steps[, i]

Appendix for code

```r
data <-read.delim("rainfall.dat", header=FALSE, sep="\n")[,1]

## 1
### a)

#Prior values
mu_0 <- mean(data)
sigma_0 <- sd(data)
tau_0 <- 5
v_0 <- 10
n <- length(data)
iterations <- 10000

#Prepping thetas for gibbs sampling
thetas_col <- c(rep(0,iterations))
thetas <- cbind(thetas_col,thetas_col)
thetas[1,] = c(mu_0, sigma_0^2)

#Function for draws for mu from the conditional posterior
mu_draw_cond_post <- function(sigma_2) {
    tau_n_2 <- 1/(n/sigma_2 + 1/tau_0^2)    #!
    raw <- (n/sigma_2) / (n/sigma_2 + 1/tau_0^2)#!
    mu_n <- raw*mu_0 + (1-raw) * mu_0
    return(rnorm(1,mu_n,sqrt(tau_n_2)))
}

#Fucntion for random inverse chi squared
randominvchisq <- function(vn,sigman,ndraw) {
    return(vn*sigman/rchisq(ndraw,vn))
}

#Function for draws for sigma from the conditional posterior
sigma_draw_cond_post <- function(mu) {
    v_n <-  n + v_0 #!
    sigma_n <- (v_0*sigma_0^2 + sum((data-mu)^2))/v_n #!
    return(randominvchisq(v_n,sigma_n,1))
}

#Function for draws from the h
gibbs_draw <- function(theta_n) {

    mu <- mu_draw_cond_post(theta_n[2])
    sigma_2 <- sigma_draw_cond_post(mu)
    return(c(mu,sigma_2))
}


#Draws for the set number of observations. First is already set above, hence we start at 2
for (i in 2:iterations) {
    thetas[i,] <- gibbs_draw(thetas[i-1,])
}
```

```r
plot(thetas[,1], type="l", main="Mu")
plot(thetas[,2], type="l", main="Sigma squared")

hist(thetas[,1], main="Mu")
hist(thetas[,2], main="Sigma squared")


### b)

##########    BEGIN USER INPUT ################
# Data options

x <- as.matrix(read.delim("rainfall.dat", header=FALSE, sep="\n"))

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

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(mu_0,nComp) # Prior mean of mu
tau2Prior <- rep(tau_0^2,nComp) # Prior std of mu
sigma2_0 <- rep(var(x),nComp) # s20 (best guess of sigma2)
nu0 <- rep(v_0,nComp) # degrees of freedom for prior on sigma2

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

# Plotting options
plotFit <- TRUE
lineColors <- c("blue", "green", "magenta", 'yellow')
sleepTime <- 0.05 # Adding sleep time between iterations for plotting
################   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)
    }
    piDraws = piDraws/sum(piDraws) # Diving every column of piDraws by the sum of the elements in that
    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)
```

```r
    for (i in 1:n){
        alloc[i] <- which(S[i,] == 1)
    }
    return(alloc)
}

# 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
    nAlloc <- colSums(S)
    #print(nAlloc)
    # 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))
    }

    # 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[al
    }

    # 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)))
    }
```

```r
    # Printing the fitted density against data histogram
    if (plotFit && (k%%1 ==0)){
        effIterCount <- effIterCount + 1

        hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = paste("Iteration number"
        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=c(



#c)
par(mfrow = c(1, 1))

hist(x, breaks = 20, freq = FALSE, xlim = c(xGridMin,xGridMax), main = "Comparison")
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "red")
lines(xGrid,dnorm(xGrid, mean(thetas[,1]), sd=mean(sqrt(thetas[,2]))), type = "l", lwd = 2, col = "orang
legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density from 1a)"
```