TDDE07 Bayesian Learning - Lab 3

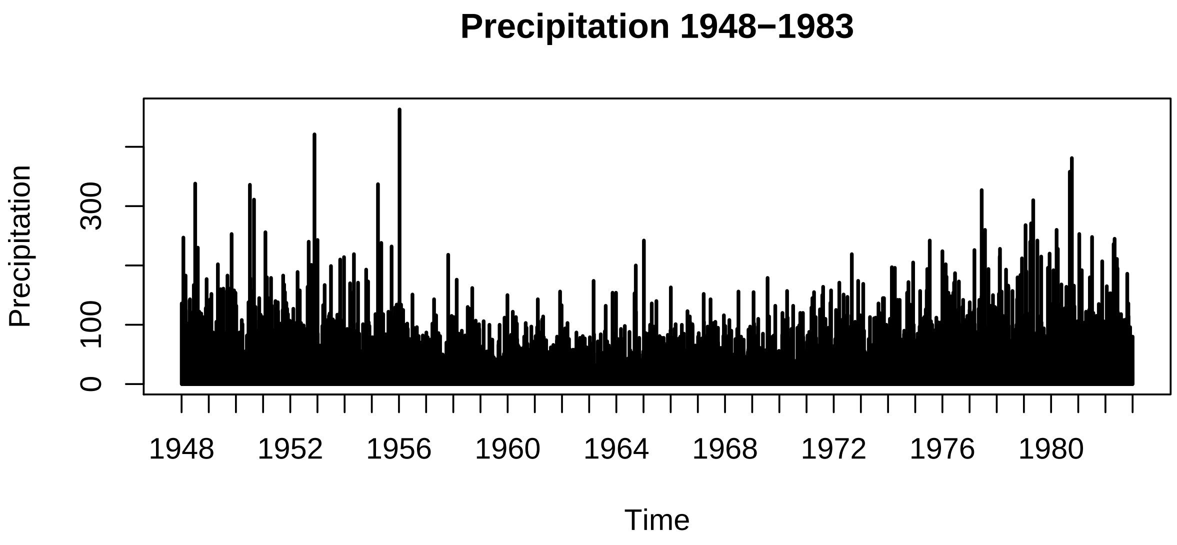
Erik Linder-Norén - erino397

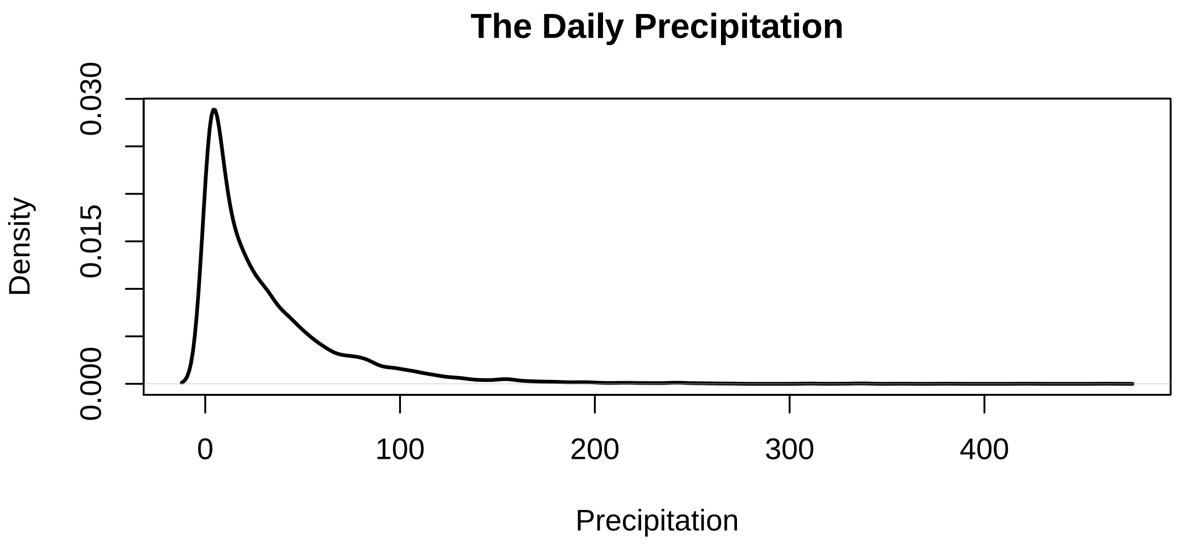
2017-05-09

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

(a) Normal model

Plotting the precipitation data and the density of that data resulted in Figure 1.





The convergence of the sampled parameters μ and σ^2 can be seen in Figure 2, where the mean of sequential draws during Gibbs sampling are calculated and visualized to show the autocorrelation between those draws, and how the sampled parameters approaches the true respective values.

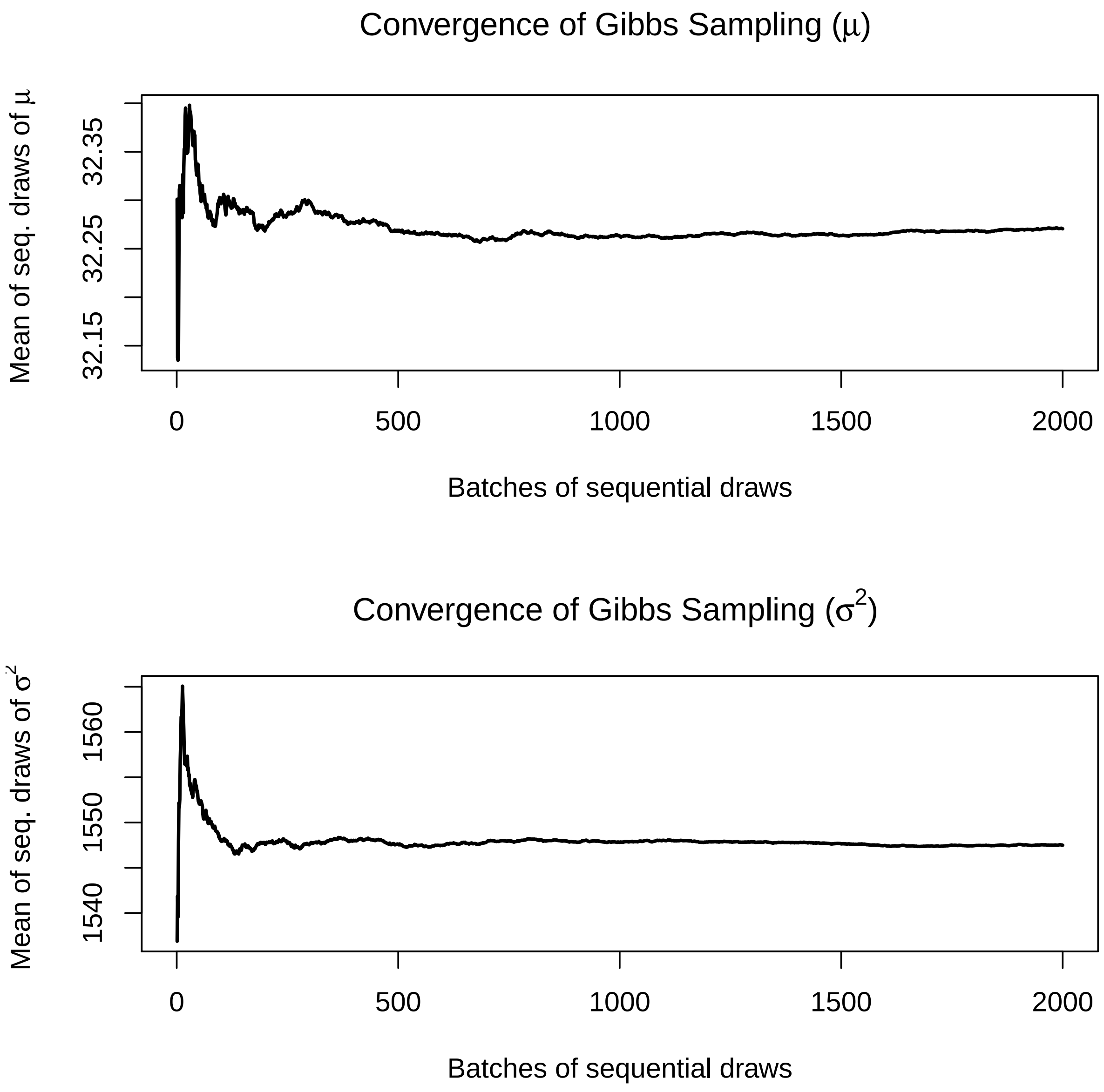


Figure 1: Predictions beta priors

(b) Mixture normal model.

The convergence of the parameters μ and σ^2 can be seen in Figure 3.

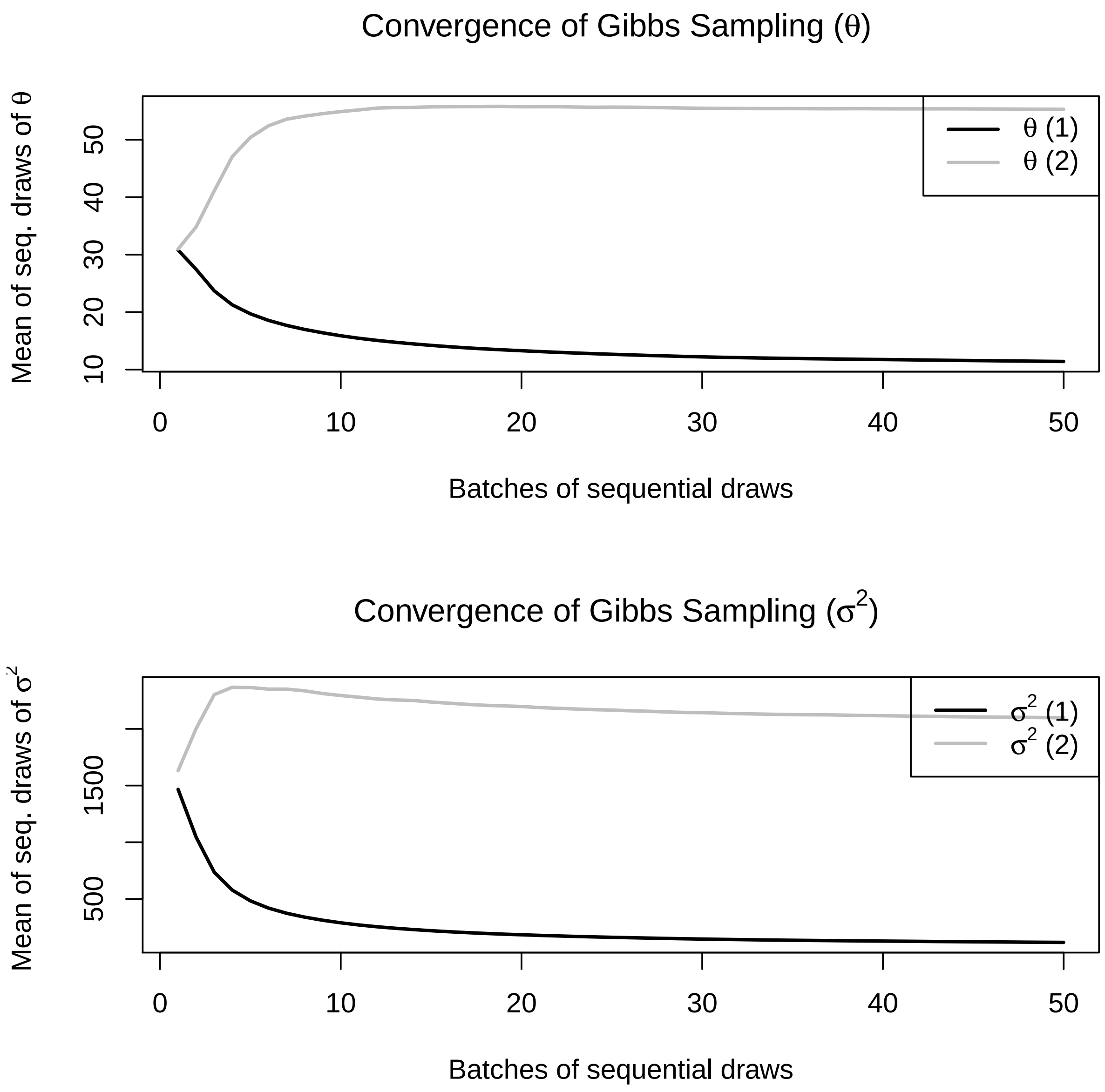


Figure 2: Mixture normal model

(c) Graphical comparison.

A comparison between the kernel density estimate of the data, a normal approximation by Gibbs sampling with parameters derived in (a) and a mixture of normals from (b) can be seen in Figure 4. Comparing a normal approximation of the data density with a mixture of normals approximation of the data - it is clear that a mixture of normals is a much better approximation of the true density of the data.

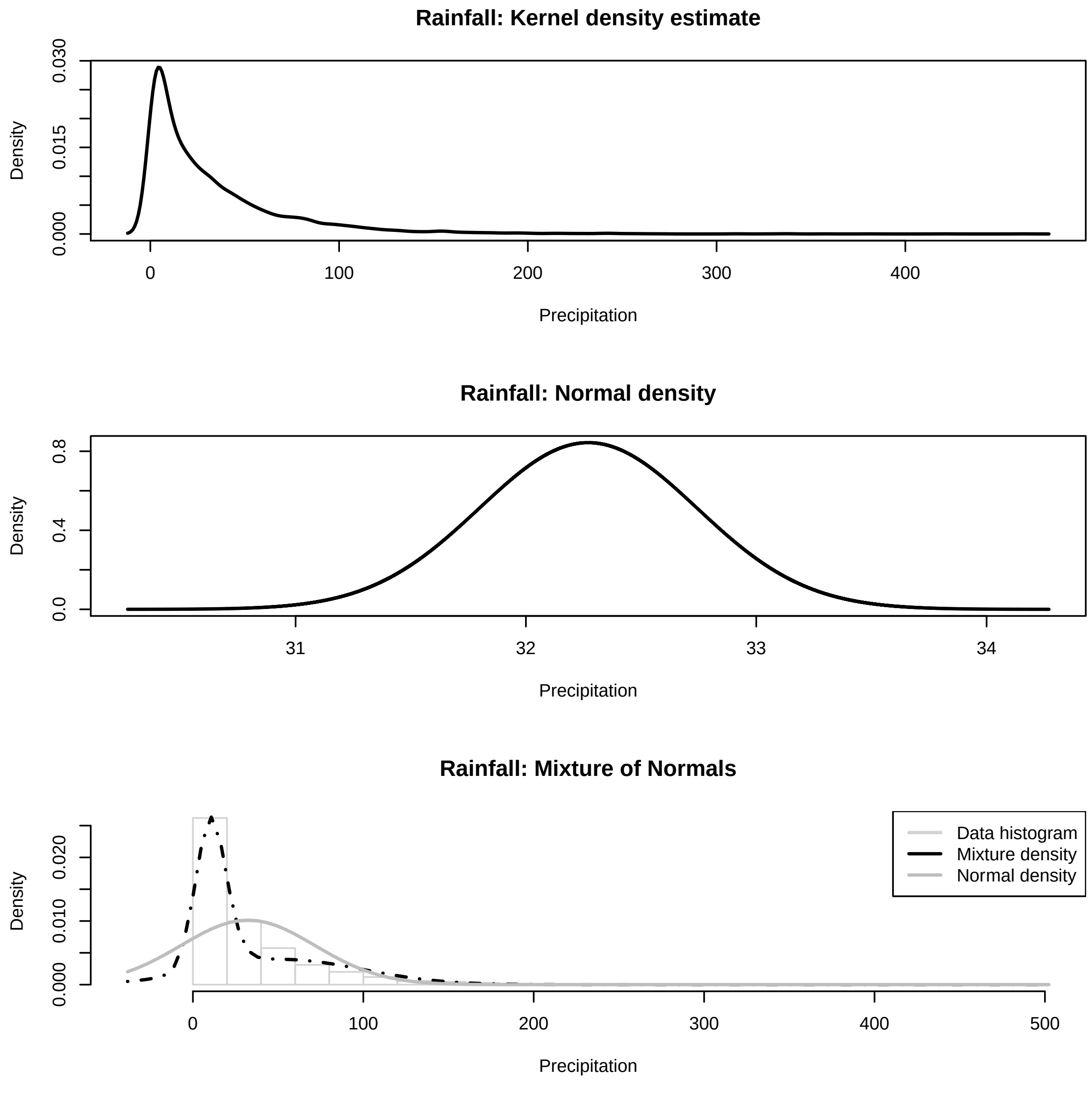


Figure 3: Mixture normal model

2. Probit regression

(c)

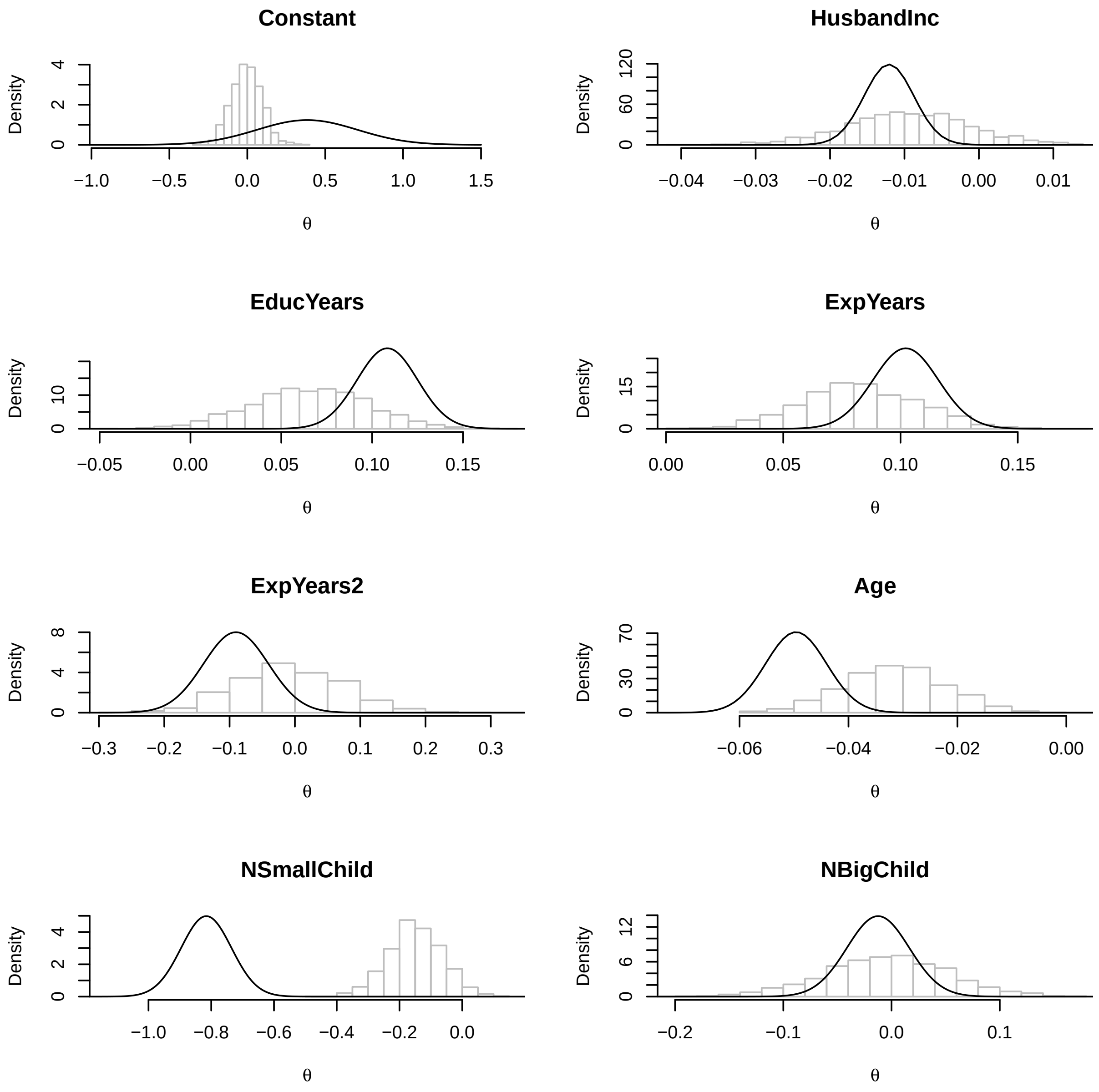


Figure 4: Comparison between the normal approximation of the probability distribution for each parameter and the histogram of the draws for those parameters during Gibbs sampling done in 1a

Assignment 1

```r
require(MASS)
require(geoR)

grid_w = 5
grid_h = 4

# Lab 3 - Assignment 1

data = read.table("data/rainfall.txt", header=FALSE)[,1]

n = length(data)

# (a)

pdf("plots/3_1_1_precipitation.pdf")

par(mfrow=c(2,1))

# Plot the precipitation
plot(data,
     type="h",
     lwd=2,
     ylab="Precipitation",
     xlab="Time",
     xaxt="n",
     col="black",
     main="Precipitation 1948-1983")

axis(1,
     at=seq(0, n, n/(1983-1948)),
     labels=seq(1948, 1983))

# Plot the precipitation density
prec_density = density(data)
plot(prec_density,
     type="l",
     lwd=2,
     xlab="Precipitation",
     ylab="Density",
     main="The Daily Precipitation")

dev.off()

data_mean = mean(data)

# Prior parameters for sigma (variance)
v0 = 1
sigma_sq0 = 1 / v0

n_draws = 4000
```

```r
# Initial value for sigma
sigma_sq = rinvchisq(n=1, v0, sigma_sq0)


gibbs_draws = matrix(0,n_draws,2)
for(i in 1:n_draws){
  mu = rnorm(n=1, mean=data_mean, sd=sqrt(sigma_sq/n))
  sigma_sq = rinvchisq(n=1, v0 + n, (v0*sigma_sq0 + sum((data - mu)^2))/(n + v0))
  gibbs_draws[i,] = c(mu, sigma_sq)
}


mean_draws = gibbs_draws[,1]
var_draws = gibbs_draws[,2]


# Calculate mean of batches of 2 draws to visualize the
# auto correlation between sequential draws
mean_means = c()
mean_vars = c()
for (i in 1:n_draws){
  if(i%%2 == 0){
    mean_means = c(mean_means, mean(mean_draws[i-1:i]))
    mean_vars = c(mean_vars, mean(var_draws[i-1:i]))
  }
}

# Plots displaying convergence of the Normal hyper
# parameters during sampling

pdf("plots/3_1_1_gibbs_sampl_conv.pdf")

par(mfrow=c(2,1))

# Plot the auto correlation (convergence) between draws of mu
min_mean = min(mean_means)
max_mean = max(mean_means)
plot(mean_means,
     type="l",
     ylim=c(min_mean, max_mean),
     cex=.1,
     lwd=2,
     main=expression(paste("Convergence of Gibbs Sampling ", "(", mu, ")", sep=" ")),
     xlab="Batches of sequential draws",
     ylab=expression(paste("Mean of seq. draws of ", mu, sep=" ")))



# Plot the auto correlation (convergence) between draws of sigma
min_var = min(mean_vars)
max_var = max(mean_vars)
plot(mean_vars,
     type="l",
     ylim=c(min_var, max_var),
     cex=.1,
     lwd=2,
     main=expression(paste("Convergence of Gibbs Sampling ", "(", sigma^2, ")", sep=" ")),
     xlab="Batches of sequential draws",
     ylab=expression(paste("Mean of seq. draws of ", sigma^2, sep=" ")))

dev.off()

# (c)

# Kernel density estimate
pdf("plots/3_1_3_dens_comp.pdf")

par(mfrow=c(3,1))

kernel_density = density(data)

plot(kernel_density$x,
     kernel_density$y,
     type="l",
     cex=.1,
     lwd=2,
     ylab="Density",
     xlab="Precipitation",
     main="Rainfall: Kernel density estimate")

# Normal density from (a)

mean = mean(mean_draws)
std_dev = sqrt(mean(var_draws)/n)

x_grid = seq(mean - 2, mean + 2, 0.0001)

normal_density = dnorm(x_grid, mean=mean, sd=std_dev)

plot(x_grid,
     normal_density,
     type="l",
     cex=.1,
     lwd=2,
     ylab="Density",
     xlab="Precipitation",
     main="Rainfall: Normal density")

# Mixture of normals from (b)
# (run './template/gaussian_mixture.R' first)

hist(x, breaks = 20, cex=.1, border="lightgray", freq = FALSE, xlim = c(xGridMin,xGridMax), xlab="Preci
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "black")
lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), type = "l", lwd = 2, col = "gray")
legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"), col=c(
dev.off()
```

Assignment 2

```r
require(mvtnorm)
require(msm)
require(MASS)
library(LaplacesDemon)

grid_w = 6
grid_h = 5

# -------
#  Lab 3
# -------


# Read data
data <- read.table("./data/WomenWork.dat", header = TRUE)

feature_labels = colnames(data)

y <- as.vector(data$Work)
X <- as.matrix(data[, 2:ncol(data)])

# Data spec.
n_features = ncol(X)
n_samples = nrow(X)

tau = 10


# -------------
#  (a) and (b)
# -------------


# Beta prior parameters
mu0 = rep(0, n_features)
covar0 = diag(tau^2, n_features)

draw_beta = function(y) {

  X_X = t(X) %*% X

  # Least squares approximate of beta
  beta_hat = ginv(X_X) %*% t(X) %*% y

  # Posterior parameters
  mu_n = ginv(X_X + covar0)%*%(X_X%*%beta_hat+covar0%*%mu0)
  covar_n = X_X + covar0

  # Assuming sigma_sq = 1
  b_draw = rmvnorm(1, mean=mu_n, sigma=ginv(covar_n))
  return(b_draw)
}

draw_u = function(beta) {

  # Mean of predictive distr.
  regr_mean = X %*% t(beta)

  u = rep(0, n_samples)
  for(i in 1:n_samples) {
    y_i = y[i]

    if(y_i == 0){
      # Truncate [-inf, 0)
      u[i] = rtnorm(n=1, mean=regr_mean[i], sd=1, upper=0)
    }else{
      # Truncate (0, inf]
      u[i] = rtnorm(n=1, mean=regr_mean[i], sd=1, lower=0)
    }
  }

  return(u)
}

# Initial prediction
u = rnorm(n_samples, covar0)

n_draws = 1500
beta_draws = matrix(0, n_draws, n_features)
u_draws = matrix(0, n_draws, n_samples)
for(i in 1:n_draws) {
  beta = draw_beta(u)
  u = draw_u(beta)
  beta_draws[i,] = beta
  u_draws[i,] = u
}

# Avoid first 10% of the draws
burn_in = floor(n_draws / 10)
beta_draws = beta_draws[burn_in:nrow(beta_draws),]

# -----
#  (c)
# -----


# Calculate the log posterior
LogPosteriorProbit <- function(betas, y, X, mu, Sigma){

  # Multiply data by parameters to get predictions
  predictions <- X%*%betas;

  # Log likelihood (for probit)
  log_likelihood = sum(y*pnorm(predictions, log.p = TRUE) + (1-y)*pnorm(predictions, log.p = TRUE, lowe

  # Log prior
  log_prior <- dmvnorm(betas, mu0, covar0, log=TRUE);

  # Sum of log likelihood and log prior is log posterior
  return(log_likelihood + log_prior)
}

log_posterior = LogPosteriorProbit

# Initialize as zeros
init_betas = rep(0, n_features)

opt_results = optim(init_betas,
                    log_posterior,
                    gr=NULL,
                    y,
                    X,
                    mu0,
                    covar0,
                    method=c("BFGS"),
                    control=list(fnscale=-1),
                    hessian=TRUE)

# Posterior mode (beta hat)
post_mode = opt_results$par
# Posterior covariance (J^-1(beta hat))
post_cov = -solve(opt_results$hessian)

pdf("plots/3_2_3_norm_gibbs_comp.pdf")

par(mfrow=c(4,2))

beta_grid = seq(-1.5, 1.5, 0.001)
for (i in 1:n_features) {

  # Build histogram of Gibbs draws of beta_i
  h = hist(beta_draws[,i], breaks=20, plot=FALSE)

  # Get the normal approximation for beta_i via optim.
  mean = post_mode[i]
  std_dev = sqrt(post_cov[i,i]/n_features)
  norm_approx = dnorm(x=beta_grid, mean=mean, sd=std_dev)

  # Find x- and y-limits for plot
  min_x = min(c(min(beta_draws[,i]), mean-4*std_dev))
  max_x = max(c(max(beta_draws[,i]), mean+4*std_dev))
  max_y = max(c(max(norm_approx), max(h$density)))

  # Plot the histogram
  plot(h,
       freq=FALSE,
       xlim=c(min_x,max_x),
       ylim=c(0,max_y),
       border="gray",
       xlab=expression(theta),
       main=feature_labels[i+1])

  # Plot the normal approximation
  lines(beta_grid, norm_approx)
}

dev.off()
```

Modifications to Mattias impl. of Mixture of Normals

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

##########    BEGIN USER INPUT #################
# Data options
data(faithful)
rawData <- faithful
x <- as.matrix(rawData['eruptions'])

# Lab 3 -
data = read.table("data/rainfall.txt", header=FALSE)[,1]
x = as.matrix(data)

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

# Prior options
alpha <- 10*rep(1,nComp) # Dirichlet(alpha)
muPrior <- rep(0,nComp) # Prior mean of theta
tau2Prior <- rep(10,nComp) # Prior std theta
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 <- 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 ###############

###### 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)
  thetaDraws <- matrix(NA,nCat,1)
  for (j in 1:nCat){
    thetaDraws[j] <- rgamma(1,param[j],1)
  }
  thetaDraws = thetaDraws/sum(thetaDraws) # Diving every column of ThetaDraws by the sum of the element
  return(thetaDraws)
}

# 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)
}

# 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
theta <- 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))

gibbs_thetas = matrix(0,nIter,2)
gibbs_sigmas = matrix(0,nIter,2)
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
  w <- rDirichlet(alpha + nAlloc)

  # Update theta'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
    theta[j] <- rnorm(1, mean = muPost, sd = sqrt(tau2Post))
  }

  gibbs_thetas[k, ] = theta

  # 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 =
  }

  gibbs_sigmas[k,] = sigma2

  # Update allocation
  for (i in 1:nObs){
    for (j in 1:nComp){
      probObsInComp[j] <- w[j]*dnorm(x[i], mean = theta[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
    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,theta[j],sd = sqrt(sigma2[j]))
      mixDens <- mixDens + w[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("topleft", box.lty = 1, legend = c("Data histogram",components, 'Mixture'),
           col = c("black",lineColors[1:nComp], 'red'), lwd = 2)
    Sys.sleep(sleepTime)
  }

}


# Calculate mean of batches of 2 draws to visualize the
# auto correlation between sequential draws
t1 = c()
t2 = c()
s1 = c()
s2 = c()
for (i in 1:nIter){
  if(i%%2 == 0){
    t1 = c(t1, mean(gibbs_thetas[,1][i-1:i]))
    t2 = c(t2, mean(gibbs_thetas[,2][i-1:i]))
    s1 = c(s1, mean(gibbs_sigmas[,1][i-1:i]))
    s2 = c(s2, mean(gibbs_sigmas[,2][i-1:i]))
  }
}


# Plots displaying convergence of the Normal hyper
# parameters during sampling

pdf("plots/3_1_2_mixt_conv.pdf")

par(mfrow=c(2,1))
```

```r
# Plot the auto correlation (convergence) between draws of mu
min_t = min(c(min(t1), min(t2)))
max_t = max(c(max(t1), max(t2)))
plot(t1,
     type="l",
     ylim=c(min_t, max_t),
     cex=.1,
     lwd=2,
     main=expression(paste("Convergence of Gibbs Sampling ", "(", theta, ")", sep=" ")),
     xlab="Batches of sequential draws",
     ylab=expression(paste("Mean of seq. draws of ", theta, sep=" ")))

lines(t2, lwd=2, col="gray")

legend("topright",
       box.lty = 1,
       legend = c(expression(paste(theta, " (1)", sep=" ")),
                  expression(paste(theta, " (2)", sep=" "))),
       col=c("black","gray"),
       lwd = 2)

# Plot the auto correlation (convergence) between draws of sigma
min_s = min(c(min(s1), min(s2)))
max_s = max(c(max(s1), max(s2)))
plot(s1,
     type="l",
     ylim=c(min_s, max_s),
     cex=.1,
     lwd=2,
     main=expression(paste("Convergence of Gibbs Sampling ", "(", sigma^2, ")", sep=" ")),
     xlab="Batches of sequential draws",
     ylab=expression(paste("Mean of seq. draws of ", sigma^2, sep=" ")))

lines(s2, lwd=2, col="gray")

legend("topright",
       box.lty = 1,
       legend = c(expression(paste(sigma^2, " (1)", sep=" ")),
                  expression(paste(sigma^2, " (2)", sep=" "))),
       col=c("black","gray"),
       lwd = 2)

dev.off()

grid_w = 6
grid_h = 5

pdf("plots/3_1_3_mixt_norm.pdf", width=grid_w, height=grid_h)

hist(x, breaks = 20, cex=.1, border="lightgray", freq = FALSE, xlim = c(xGridMin,xGridMax), xlab="Preci
lines(xGrid, mixDensMean, type = "l", lwd = 2, lty = 4, col = "black")
lines(xGrid, dnorm(xGrid, mean = mean(x), sd = apply(x,2,sd)), type = "l", lwd = 2, col = "gray")
legend("topright", box.lty = 1, legend = c("Data histogram","Mixture density","Normal density"), col=c(
dev.off()

#########################    Helper functions    ##############################################
```