Lab 1

Question 1

In this assignment we are given data coming from a Bernoulli distribution with unknown parameter θ and we use the conjugate prior

$$\theta \sim \text{Beta}(\alpha_0, \beta_0)$$

where $\alpha_0 = \beta_0 = 2$. The sample we get consists of 14 successes out of 20 trials which gives us the posterior distribution

$$\theta|y \sim \text{Beta}(\alpha_0 + 14, \beta_0 + 6).$$

```
n <- 20
s <- 14
f <- n - s

alpha_prior <- 2
beta_prior <- 2
```

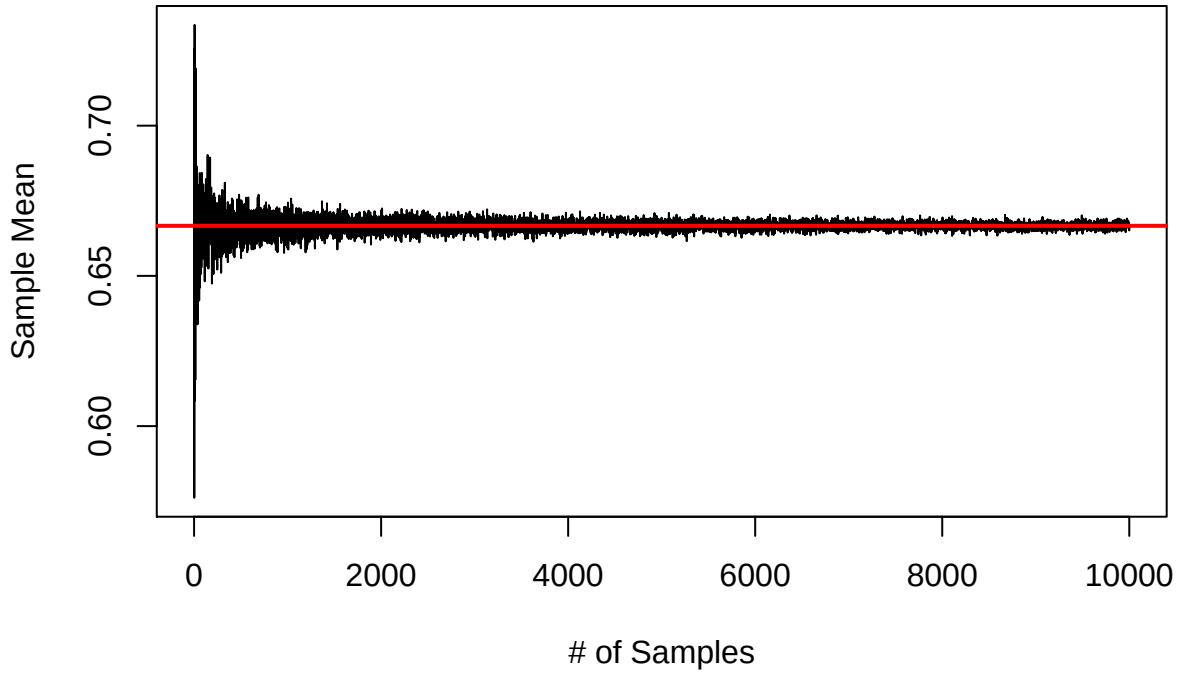
a

Here we are sampling from the posterior distribution to verify that as the number of samples increases, the mean and standard deviation converges to the true values. In this particular case the true mean is $\alpha/(\alpha+\beta) = 2/3$ and standard deviation is $\sqrt{(\alpha\beta)/((\alpha+\beta)^2(\alpha+\beta+1))} = \sqrt{2/225}$.

```
drawSamples <- function(nDraws, alpha, beta) {
    rbeta(nDraws, shape1=alpha, shape2=beta)
}

nDraws <- 10000
alpha <- alpha_prior + s
beta <- beta_prior + f

set.seed(123)
samples <- drawSamples(nDraws, alpha, beta)

true_mean <- alpha / (alpha + beta)
estimated_mean <- mean(samples)

true_sd <- sqrt((alpha * beta) / ((alpha + beta)^2 * (alpha + beta + 1)))
estimated_sd <- sd(samples)

set.seed(123)
betamean <- sapply(1:nDraws, FUN = function(X) {
    mean(drawSamples(nDraws = X, alpha = alpha , beta = beta))
})

set.seed(123)
betasd <- sapply(1:nDraws, FUN = function(X) {
```

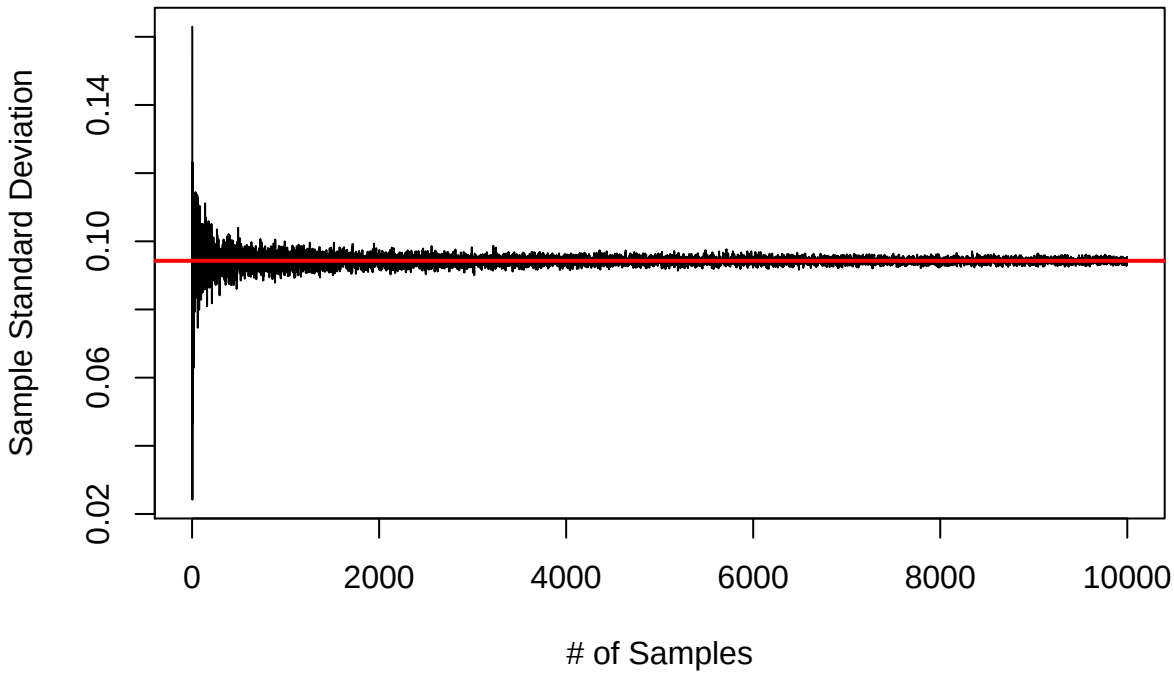
```
    sd(drawSamples(nDraws = X, alpha = alpha , beta = beta))
})
```

```
plot(x = 1:nDraws, y = betamean, type = "l",
     xlab="# of Samples", ylab="Sample Mean")
abline(h = true_mean, col = "red", lwd = 2)
```



The plot above shows how the posterior mean changes with the number of drawn samples. The red line is the expected mean and we can see that as the number of samples increase it converges to the true mean.

```
plot(x = 1:nDraws, y = betasd, type = "l",
     xlab="# of Samples", ylab="Sample Standard Deviation")
abline(h = true_sd, col = "red", lwd = 2)
```



Similar, the plot above shows how the standard deviation from the posterior changes with the number of drawn samples. The red line shows the expected standard deviation and we can see that the posterior standard deviation converges to it.

b

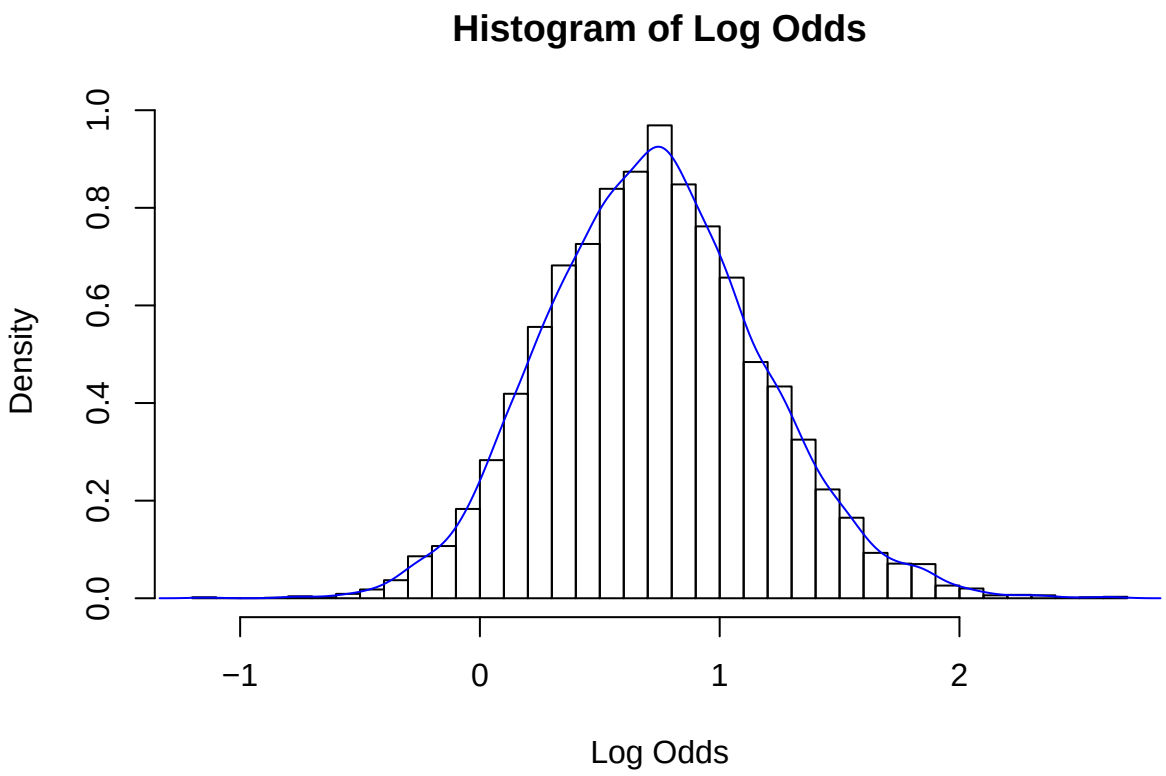
```
est_04 <- sum(samples < 0.4) / length(samples)
true_04 <- pbeta(0.4, shape1 = alpha, shape2 = beta)
```

The posterior probability for theta being below 0.4 is calculated as the share of values under 0.4 in our drawn samples. We drew 10000 samples and got $\Pr(\theta < 0.4|y) \approx 0.004$ while the true value is $\Pr(\theta < 0.4|y) = 0.004$ which is fairly close.

c

We can also use the samples to get the distribution of a function of those samples. In this case we take $\text{logit}(\theta) = \log(\theta/(1-\theta))$ and get the histogram below.

```
log_odds <- function(theta){
  log(theta / (1 - theta))
}

sample_odds <- log_odds(samples)

hist(sample_odds, breaks=50, freq = FALSE, main="Histogram of Log Odds", xlab="Log Odds")
lines(density(sample_odds), col = "blue")
```


Histogram of Log Odds
Density
0.0
0.2
0.4
0.6
0.8
1.0
−1
0
1
2
Log Odds

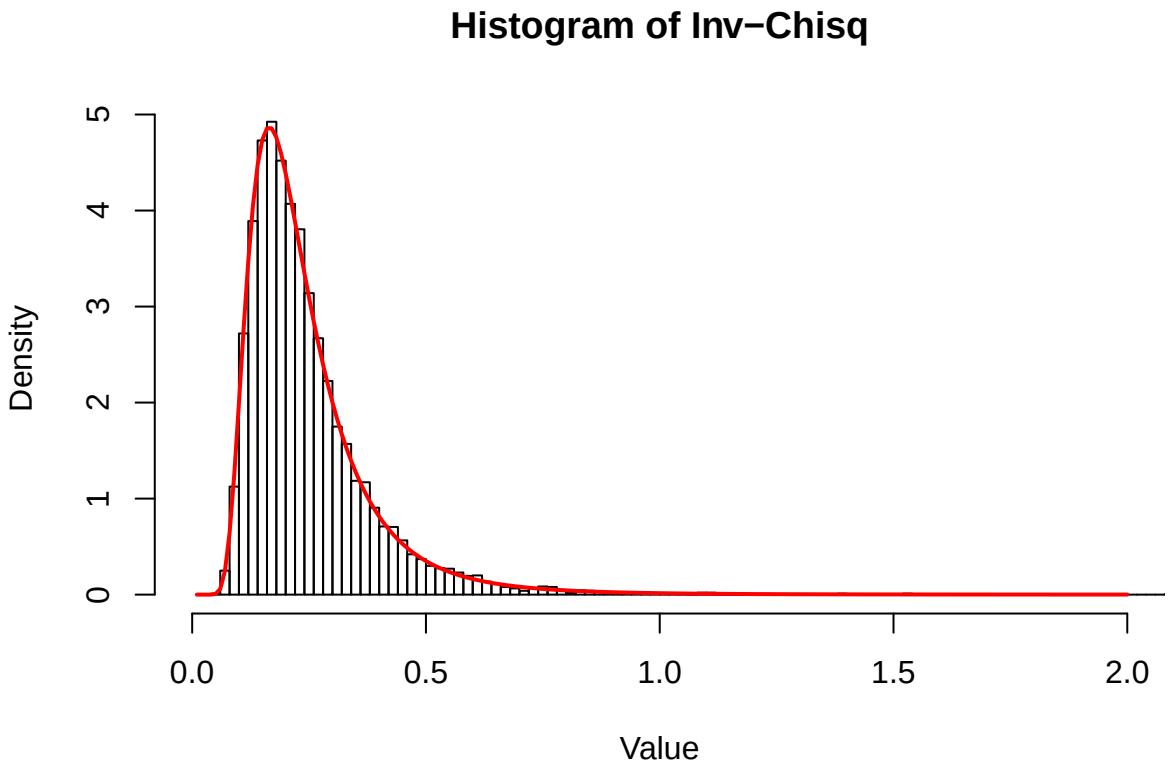
Question 2

a

```
library(geoR)
sek <- c(14, 25, 45, 25, 30, 33, 19, 50, 34 ,67)

n <- length(sek)
mu <- 3.5
tausq <- sum((log(sek)- mu)^2) / n

set.seed(123)
chi_samples <- rinvchisq(nDraws, df = n, scale = tausq)
```

```
invchisq_pdf <- function(x, df, scale) {
    factor1 <- (scale * df / 2)^(df / 2) / gamma(df / 2)
    factor2 <- exp(-(df * scale) / (2 * x)) / x^(1 + df / 2)
    factor1 * factor2
}

xs <- seq(0.01, 2, 0.01)
ys <- invchisq_pdf(xs, 10, tausq)
```

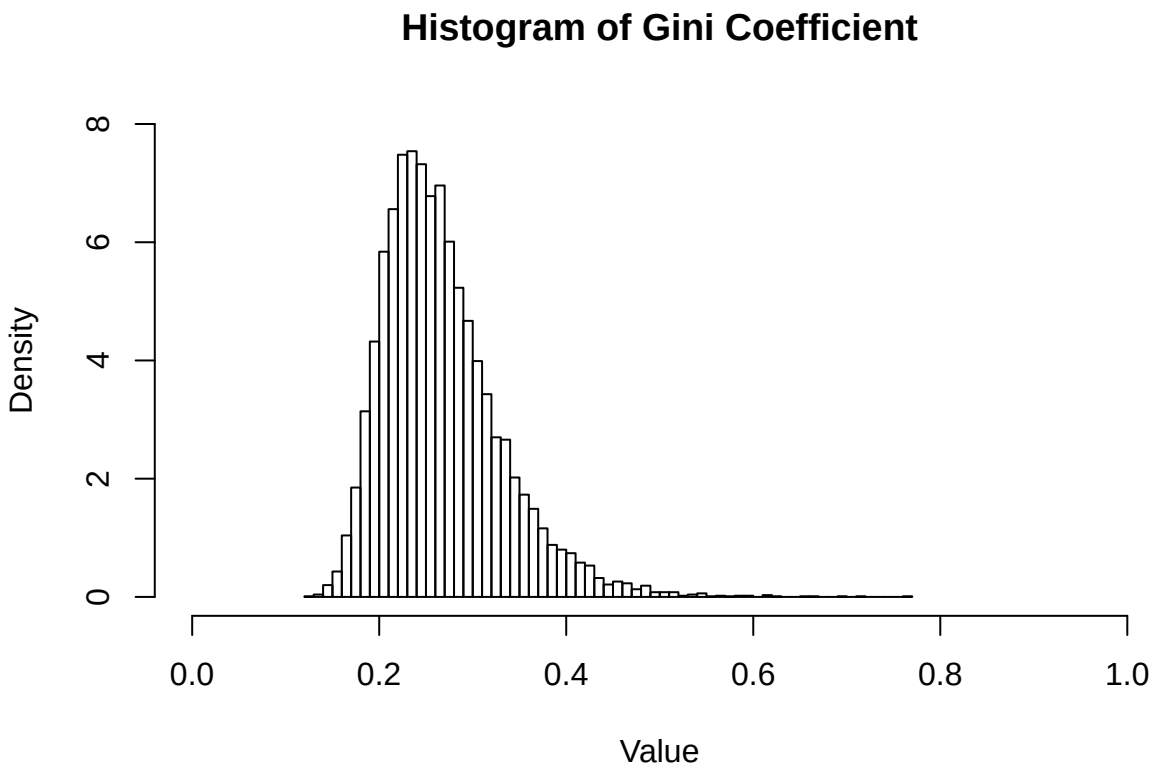
```
hist(chi_samples, breaks = 100, freq=FALSE, xlim=c(0, 2),
     main="Histogram of Inv-Chisq", xlab="Value")
lines(xs, ys, col="red", lwd=2)
```



The red line represents the theoretical distribution and it looks like a really nice fit to the posterior in the histogram.

b

```
gini <- function(sigma){
    2 * pnorm(sigma / sqrt(2), mean = 0 , sd = 1) - 1
}

gini_samples <- gini(sqrt(chi_samples))

hist(gini_samples, breaks = 50, freq = FALSE,
     ylim = c(0,8), xlim=c(0, 1), main="Histogram of Gini Coefficient", xlab="Value")
```



The histogram above shows the posterior distribution of the Gini coefficient.

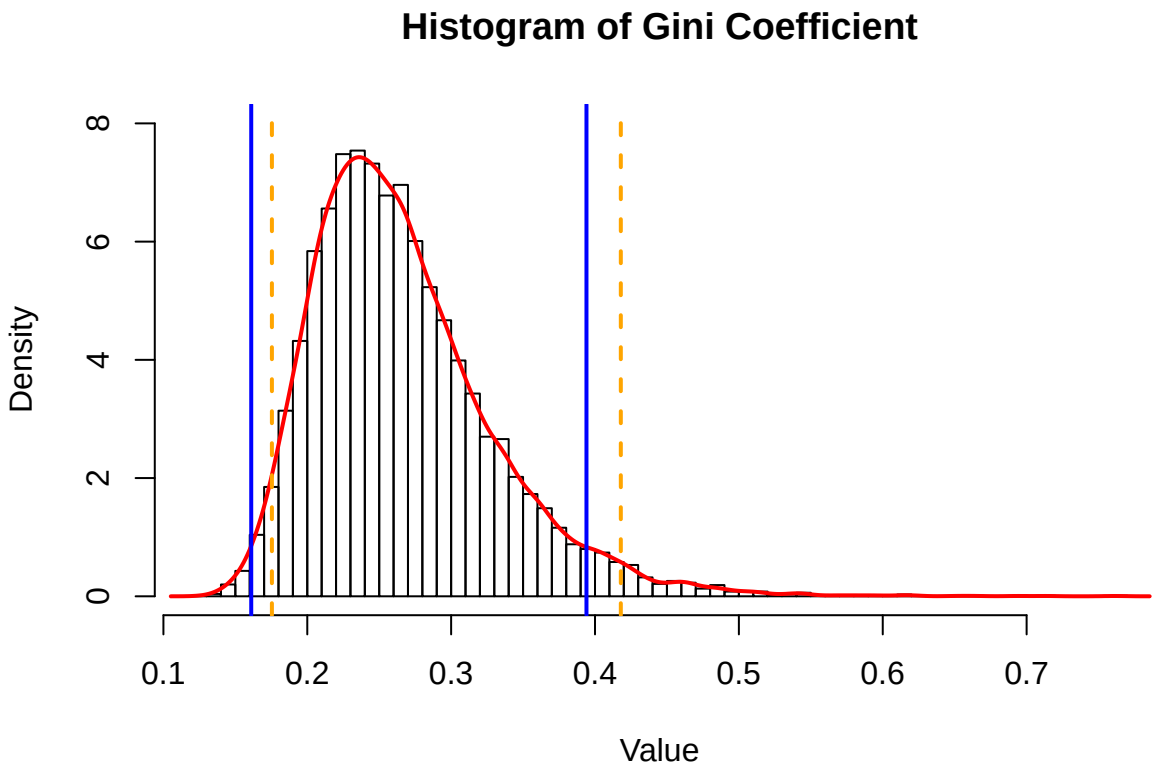
c

```
gini_cred <- quantile(gini_samples, probs = c(0.025, 1 - 0.025))
```

```
dense_gini <- density(gini_samples)
dense_gini_cdf <- dense_gini$y/sum(dense_gini$y)
index_gini <- order(dense_gini_cdf)
dense_gini_cdf <- dense_gini_cdf[index_gini]

while(sum(dense_gini_cdf) > 0.95){
    dense_gini_cdf <- dense_gini_cdf[-1]
    index_gini <- index_gini[-1]
}

lower <- min(dense_gini$x[index_gini])
```

```r
upper <- max(dense_gini$x[index_gini])

gini_HPD <- c(lower, upper)
```

```r
hist(gini_samples, breaks = 50, freq = FALSE, ylim = c(0, 8),
     main="Histogram of Gini Coefficient", xlab="Value")
lines(density(gini_samples), col = "red", lwd = 2)
abline(v = gini_HPD, col = "blue", lwd = 2)
abline(v = gini_cred, col = "orange", lwd = 2, lty = 2)
```



The credible-interval is represented by the dashed orange line and the parameter G lies within this interval with 0.95 probability and the probability of G being outside the interval is equally likely on both sides.

The Highest Posterior Density (HPD) interval contains 95% of the most probable G values.

The main difference between the two intervals is that the HPD is more shifted to the left since the distributions is positively skewed.

What we can say about the data is that it is more probable that the income-distribution is equal than inequal.

Question 3

For this exercise we have been given the likelihood

$$p(y|\mu,\kappa) = \frac{\exp\left[\kappa\cos(y-\mu)\right]}{2\pi I_0(\kappa)}, -\pi \leq y \leq \pi$$

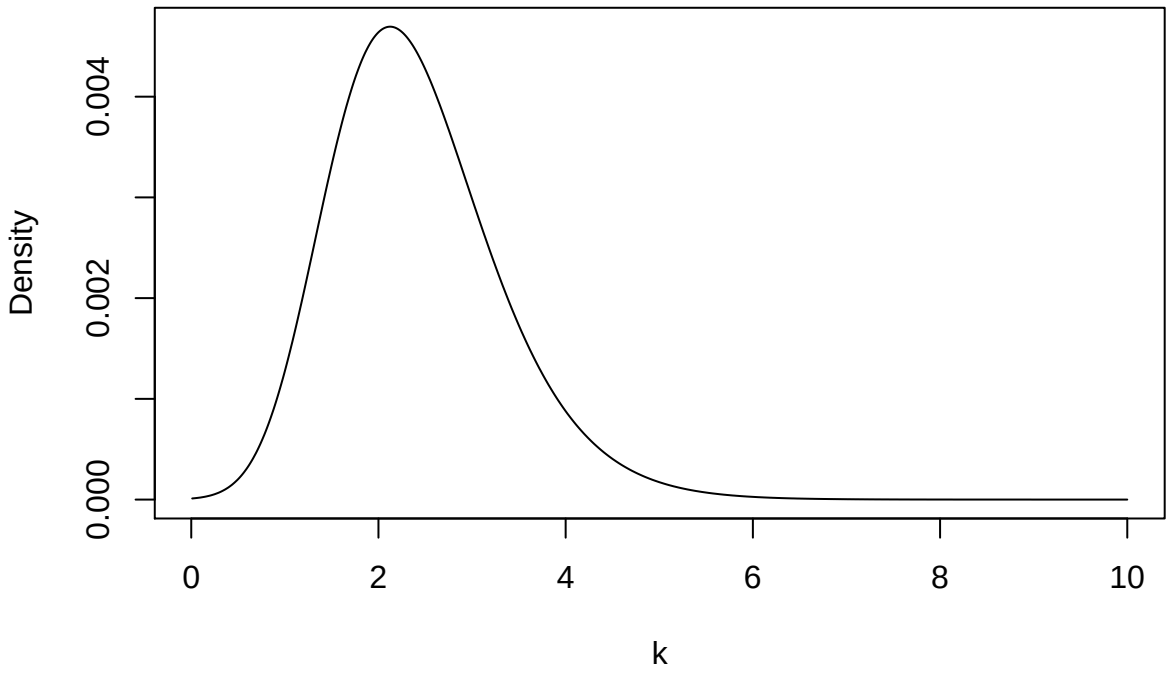
where $I_0(\kappa)$ is the modified Bessel function of the first kind of order zero. We have also been given that $\kappa \sim \text{Exponential}(\lambda = 1)$. We want to approximate the marginal posterior distribution $p(\kappa|\mu, y)$ and we assume that $\mu = 2.39$. In order to this we generate samples of κ that we assume comes from the Exponential distribution, in our case 1000 values equally spaces between 0.01 to 10. We can then compute

$$p(\kappa|\mu, y) \propto p(y|\mu,\kappa)p(\kappa)$$

and get an approximation of the marginal posterior distribution. This method usually goes by the name of *grid approximation* and is appropriate for low number of parameters.

a

```
vonMises <- function(y,K,mu){
    exp(K * cos(y - mu)) /(2 * pi * besselI(K, nu = 0))
}

y <- c(-2.44, 2.14, 2.54, 1.83, 2.02, 2.33, -2.79, 2.23, 2.07, 2.02)

sample_exp <- seq(0.01, 10, 0.01)
sample_mises <- t(sapply(y, function(X) vonMises(y = X, K = sample_exp, mu = 2.39)))

prior <- dexp(sample_exp, rate = 1)
likelihood <- apply(sample_mises, MARGIN = 2, FUN = prod)

unnorm_posterior <- likelihood * prior
posterior <- unnorm_posterior / sum(unnorm_posterior)

plot(sample_exp, posterior, type="l", ylab="Density", xlab="k")
```



We can see from the distribution above that the most density is located between 0.5 and 4.

b

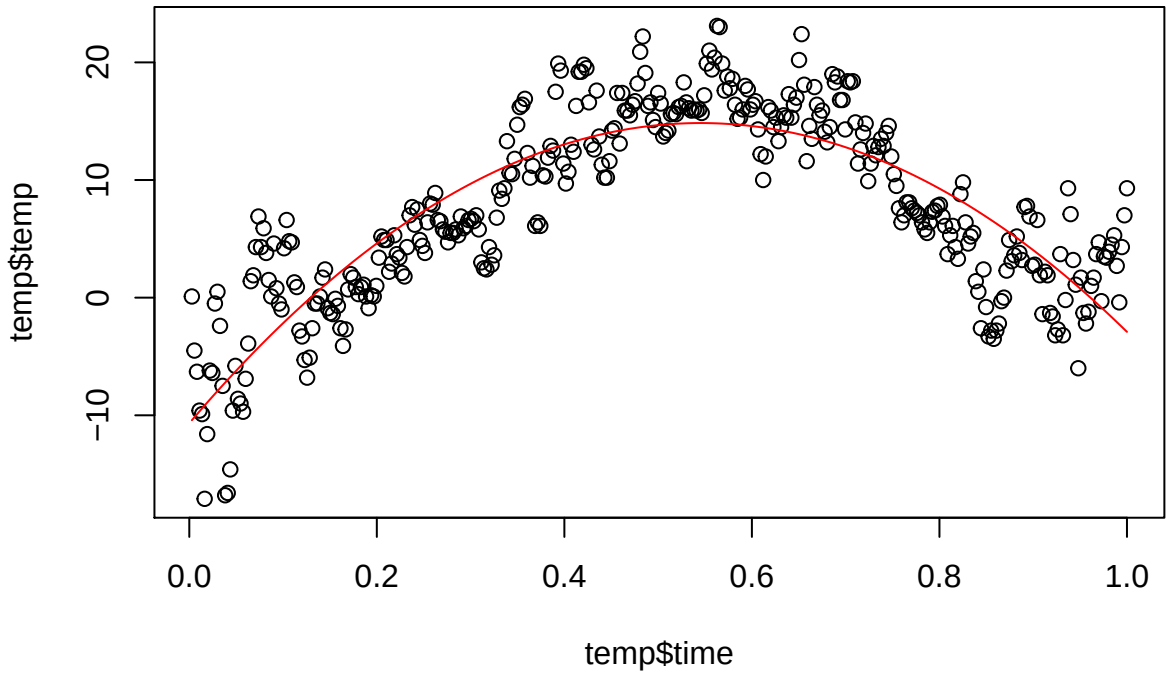
The posterior mode of κ is approximately 2.12 given by our approximation of the distribution.

Lab 2

Question 1

```
temp <- read.table("../data/TempLinkoping2016.txt", header=T)
mod <- lm(temp ~ time + I(time^2), data=temp)
```

```
idx <- order(temp$time)
x <- temp$time[idx]
y <- fitted(mod)[idx]

plot(temp$time, temp$temp)
lines(x, y, col='red', type='l')
```



Prior

$$\sigma^2 \sim \text{Inv} - \chi^2(\nu_0, \sigma_0^2)$$
$$\beta|\sigma^2 \sim \text{N}(\mu_0, \sigma^2\Omega_0^{-1})$$

Likelihood

$$\mathbf{y}|\beta, \sigma^2, \mathbf{X} \sim \text{N}(\mathbf{X}\beta, \sigma^2 I_n)$$

Posterior

$$\sigma^2|\mathbf{y} \sim \text{Inv} - \chi^2(\nu_n, \sigma_n^2)$$
$$\beta|\sigma^2, \mathbf{y} \sim \text{N}(\mu_n, \sigma^2\Omega_n^{-1})$$

where

$$\begin{aligned}
\mu_n &= (\mathbf{X}^\intercal\mathbf{X} + \Omega_0)^{-1}(\mathbf{X}^\intercal\mathbf{X}\hat{\beta} + \Omega_0\mu_0) \\
\Omega_n &= \mathbf{X}^\intercal\mathbf{X} + \Omega_0 \\
\nu_n &= \nu_0 + n \\
\nu_n\sigma_n^2 &= \nu_0\sigma_0^2 + (\mathbf{y}^\intercal\mathbf{y} + \mu_0^\intercal\Omega_0\mu_0 - \mu_n^\intercal\Omega_n\mu_n)
\end{aligned}$$

a)

```
mu0 <- c(0, 0, 0)
omega0 <- diag(3) * 0.05
nu0 <- 1
sigmasq0 <- 20

hyperparams <- list(mu=mu0, omega=omega0, nu=nu0, sigmasq=sigmasq0)
```

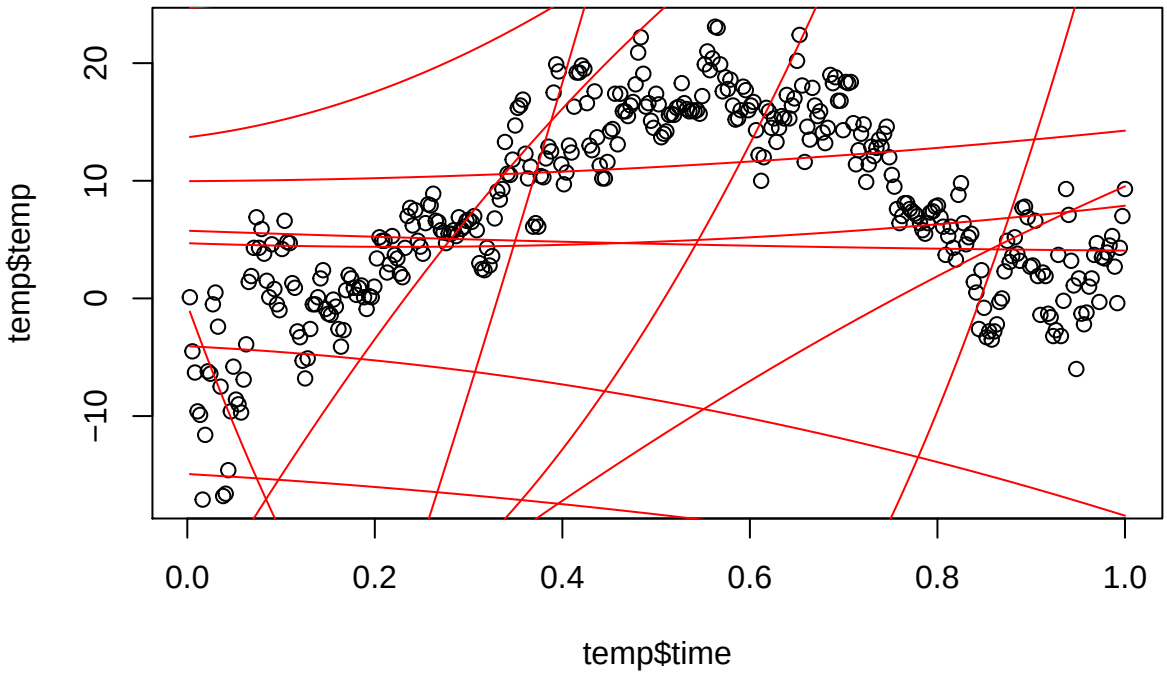
Since we both are novice in weather-prediction we have no real prior knowledge we set all mu0 to 0. Our omega0 is set to a diagonal matrix with 0.05 in the trace to express that we are not certain at all in these mus.

The same goes for our priors for sigma, with nu0 (τ_0) we express that we are very uncertain of our set prior for sigmasq.

b)

```
library(geoR)
library(MASS)

time <- data.frame(rep(1,nrow(temp)), temp$time, temp$time^2)
mtime <- as.matrix(time)
mtemp <- matrix(temp$temp, ncol = 1)

prior_estimate <- function(data, params) {
    sigmasq  <- rinvchisq( n = 1, df = params$nu, scale = params$sigmasq)
    betacoef <- mvrnorm(n = 1, mu = params$mu, Sigma = sigmasq * solve(params$omega) )

    data %*% betacoef
}
```

```
plot(temp$time, temp$temp)

x <- sort(temp$time)

set.seed(12345)

for (i in 1:20){
    y <- prior_estimate(mtime, hyperparams)[order(temp$time)]
    lines(x, y, col='red', type='l')
}
```



Given our prior the curves are very flexible and go all over the plot which is about what we expected when setting such vauge priors.

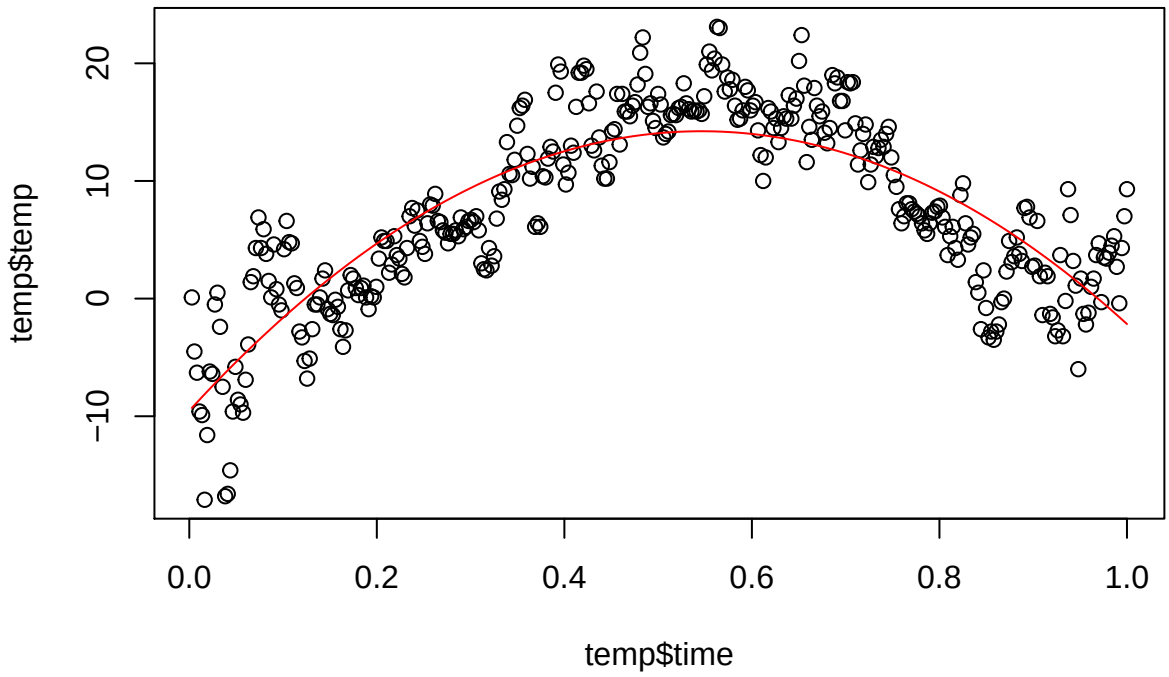
Since we say that we don't really know anything about the weather-forecast we are satitisfied with this.

c)

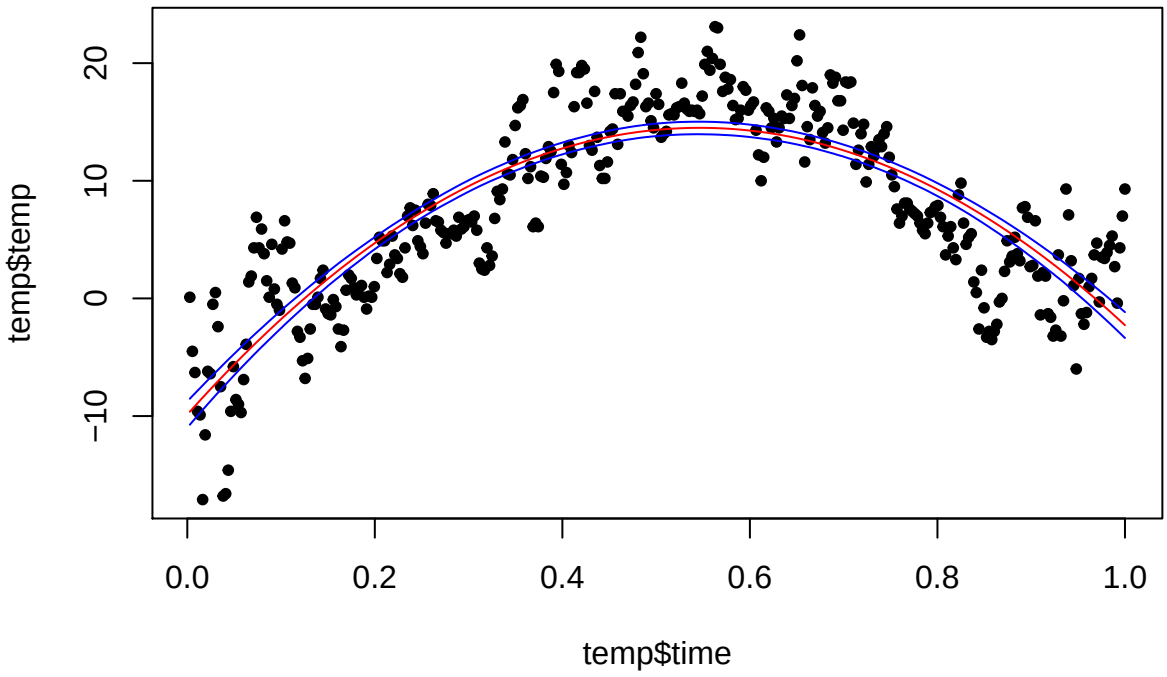
```
posterior_param_sample <- function(X, y, hyperparams){
    XX <- t(X) %*% X

    betahat <- solve(XX) %*% t(X) %*% y

    mun <- solve(XX + hyperparams$omega) %*%
        (XX %*% betahat + hyperparams$omega %*% hyperparams$mu)

    omegan <- XX + hyperparams$omega

    nun <- hyperparams$nu + nrow(X)

    nunsigmasqn <- hyperparams$nu * hyperparams$sigmasq +
        (t(y) %*% y +
         t(hyperparams$mu) %*% hyperparams$omega %*% hyperparams$mu -
         t(mun) %*% omegan %*% mun )

    sigmasqn <- nunsigmasqn / nun

    sigmasq  <- rinvchisq(n = 1, df=nun, scale=sigmasqn)
    beta <- mvrnorm(n = 1, mu = mun, Sigma = as.numeric(sigmasq) * solve(omegan))

    list(beta = beta, sigmasq = sigmasq)
}
```

```
posterior_estimate <- function(X, y, hyperparams){
    sample <- posterior_param_sample(X, y, hyperparams)
    X %*% sample$beta
}
```

```
plot(temp$time, temp$temp)

set.seed(12345)

idx <- order(temp$time)
x <- temp$time[idx]
y <- posterior_estimate(mtime, mtemp, hyperparams)[idx]

lines(x, y, col='red', type='l')
```




```
set.seed(12345)
library(fields)
ests <- sapply(1:1000, FUN = function(x) posterior_estimate(mtime, mtemp, hyperparams))
cred_interval <- apply(ests, MARGIN = 1, quantile, probs = c(0.05, 0.95))

idx <- order(temp$time)
x <- temp$time[idx]
y1 <- rowMeans(ests)[idx]
y2 <- cred_interval[1,][idx]
y3 <- cred_interval[2,][idx]

plot(temp$time, temp$temp, pch = 20 )
lines(x, y1, col='red', type='l')
lines(x, y2, col='blue', type='l')
lines(x, y3, col='blue', type='l')
```



d)

```
set.seed(12345)

betas <- sapply(1:1000, FUN = function(x) posterior_param_sample(mtime, mtemp, hyperparams)$beta)
hot <- mean(-betas[2,] / (2 * betas[3,]))
hot * 366 # July 27, 2016 (Wed)
```

```
## [1] 200
```

e)

Set μ_0 to zeros for all of them. For Ω_0 the first three elements in the diagonal are set low and the later ones set high. The prior now express that we are uncertain of what the first three mus are and that we are certain that the other five terms are the mus specified (i.e zero, not needed).

Question 2

```
women <- read.table("../data/WomenWorks.txt", header = TRUE)
```

a)

```
glmModel <- glm(Work ~ 0 + ., data = women, family = binomial)
```

b)

```
library(mvtnorm)

logprior <- function(beta, mean, sigma){
    dmvnorm(beta, mean = mean, sigma = sigma, log = TRUE)
}

loglikelihood <- function(beta, X, Y){
    linear_prediction <- t(X) %*% beta

    probabilities <- (Y * linear_prediction) - log(1 + exp(linear_prediction))
    loglike <- sum(probabilities)

    ## if (abs(loglike) == Inf)
    ##     loglike = -20000

    loglike
}

logposterior <- function(beta, X, Y, mean, sigma){
   loglikelihood(beta, X, Y) + logprior(beta, mean, sigma)
}

tau <- 10
mu <- rep(0,8)
sigma <- tau^2 * diag(8)

womenX <- as.matrix(women[,2:ncol(women)])
womenY <- as.matrix(women[,1])

optpost <- optim(par = matrix(rep(0, 8), ncol = 1),
                 fn = logposterior, method = "BFGS", hessian = TRUE,
                 X = t(womenX), Y = womenY,
                 mean = mu, sigma = sigma,
                 control=list(fnscale=-1))
```

```
posterior_beta_sample <- function(n, mu, sigma){
    rmvnorm(n, mean = mu, sigma = sigma)
}

mu <- optpost$par
```

```
sigma <- -solve(optpost$hessian)

set.seed(12345)
beta_samples <- posterior_beta_sample(n=1000, mu=mu, sigma=sigma)
cred_intervals <- apply(beta_samples, 2, quantile, prob=c(0.025, 0.975))
colnames(cred_intervals) <- colnames(women)[-1]
```

The 95% credibility intervals below show that the intercept, husband income, transformed years in work experience, and the number of children older than 6 years are statistical insignificant, i.e. 0 (no impact) is contained in the intervals. NsmallChilds credibility interval is not crossing 0 and therefor considerd significant and important for the model.

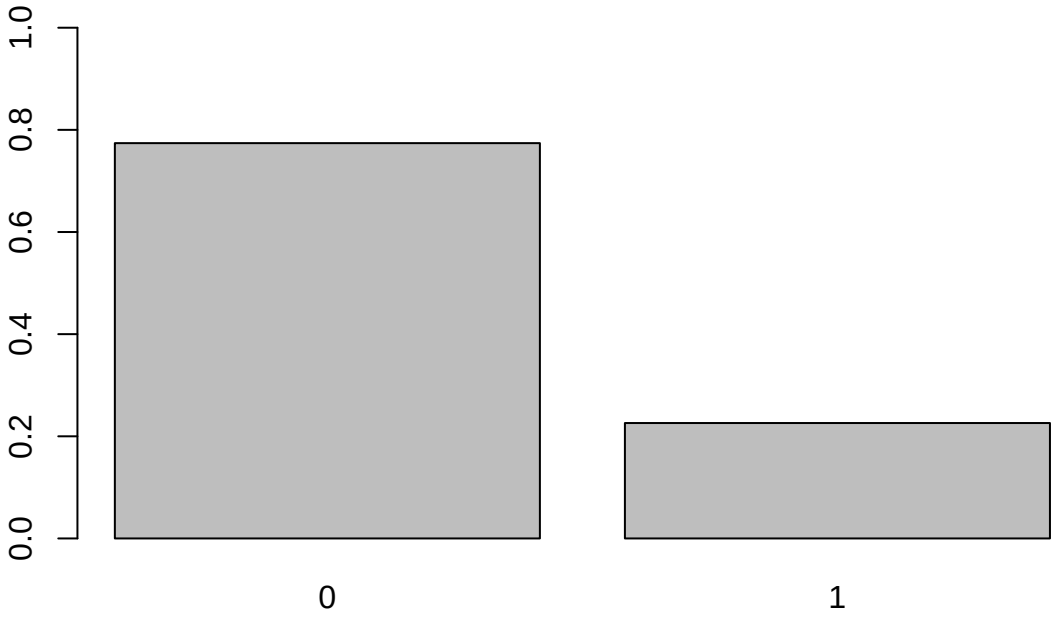
```
cred_intervals
```

```
##       Constant HusbandInc EducYears ExpYears ExpYears2    Age NSmallChild
## 2.5%     -2.31   -0.05092     0.031   0.0472    -0.614 -0.135      -2.130
## 97.5%     3.53    0.00781     0.327   0.2966     0.294 -0.031      -0.576
##       NBigChild
## 2.5%     -0.302
## 97.5%     0.269
```

c)

```
posterior_predictive_sample <- function(X, beta){
    linear_prediction <- t(X) %*% beta
    probability <- exp(linear_prediction) / (1 + exp(linear_prediction))
    rbinom(n=1, size=1, prob=probability)
}

x <- matrix(c(1, 10, 8, 10, (10 / 10)^2, 40, 1, 1), ncol=1)
prediction_samples <- apply(beta_samples, 1,  function(beta) posterior_predictive_sample(x, beta))
```

```
counts <- table(prediction_samples)
barplot(counts / sum(counts), ylim=c(0, 1))
```



The barplot above indicates that it is much more probable that given the data the woman is not working under our model.

Lab 3

Question 1

```
options(digits=2)

rainfall <- read.table("../data/rainfall.dat", header=F)
```

Prior

$$\mu \sim N(\mu_0, \tau_0^2)$$
$$\sigma^2 \sim \text{Inv}\text{–}\chi^2(\nu_0, \sigma_0^2)$$

Likelihood

$$\mathbf{y}|\mu, \sigma^2 \sim N(\mu, \sigma^2)$$

Posterior

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

where

$$\mu_n = \frac{\frac{1}{\tau_0^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}}\mu_0 + \frac{\frac{n}{\sigma^2}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma^2}}\bar{x}$$
$$\frac{1}{\tau_n^2} = \frac{1}{\tau_0^2} + \frac{n}{\sigma^2}$$
$$\nu_n = \nu_0 + n$$

a)

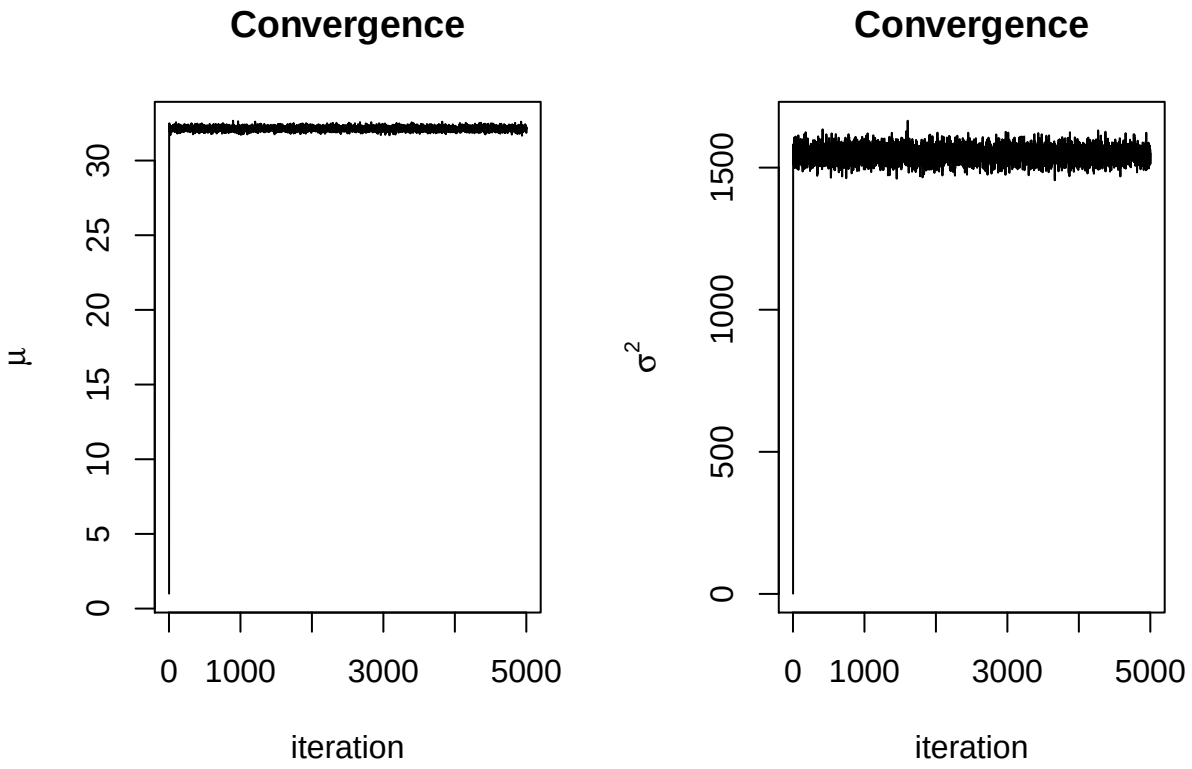
```
library(geoR)

mun <- function(x, sigmasq, hyperparams){
    n <- length(x)
    denom <- ((1 / hyperparams$tausq0) + (n / sigmasq))

    pt1 <- ((1 / hyperparams$tausq0) / denom)  * hyperparams$mu0
    pt2 <- ((n / sigmasq) / denom) * mean(x)

    pt1 + pt2
```

```
}


taun <- function(x, sigmasq, hyperparams){
    hyperparams$tausq0 + (sigmasq / length(x))
    1 / hyperparams$tausq0
}

nun <- function(x, hyperparams){
    hyperparams$nu0 + length(x)
}


sigmasqn <- function(x, mu, hyperparams){
    (hyperparams$nu0 * hyperparams$sigmasq0 + sum((x - mu)^2 )) / (length(x) + hyperparams$nu0)
}


musampler <- function(x,sigmasq, hyperparams){
    mu <- mun(x, sigmasq, hyperparams)
    sigma <- sqrt(taun(x,sigmasq, hyperparams))
    rnorm(1, mu, sigma)
}

sigmasampler <- function(x, mu, hyperparams){
    scale  <- sigmasqn(x, mu, hyperparams)
    df <- nun(x, hyperparams)
    rinvchisq(1, df, scale)
}

gibbs <- function(x, iter, init, hyperparams){
    samples <- matrix(NA, ncol = 2, nrow = iter + 1)
    samples[1,] <- init

    for (i in 2:(iter+1)){
        mu <- musampler(x, samples[i-1, 2], hyperparams)
        sigma <- sigmasampler(x, mu, hyperparams)
        samples[i,] <- c(mu, sigma)
    }

    colnames(samples) <- c("mu", "sigmasq")
    samples
}
```

```
set.seed(123456)

hyperparams <- list(mu0=0, tausq0=50, nu0=1, sigmasq0=50)
iter <- 5000

gibbs_params <- gibbs(x=rainfall$V1, iter = iter, init = c(1,1), hyperparams)
```

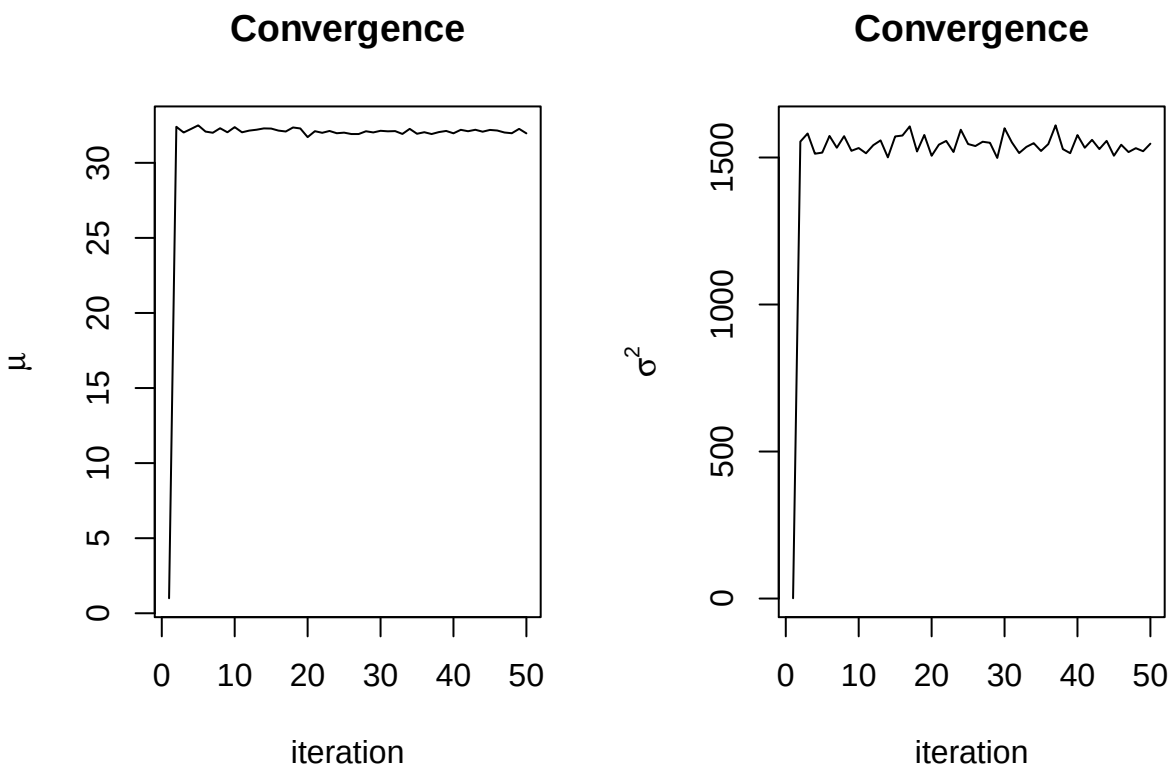
```
par(mfrow=c(1, 2))
plot(gibbs_params[, "mu"], type = "l", xlab="iteration",
     ylab=expression(mu), main="Convergence")
```

```r
plot(gibbs_params[, "sigmasq"], type = "l", xlab="iteration",
     ylab=expression(sigma^2), main="Convergence")
```



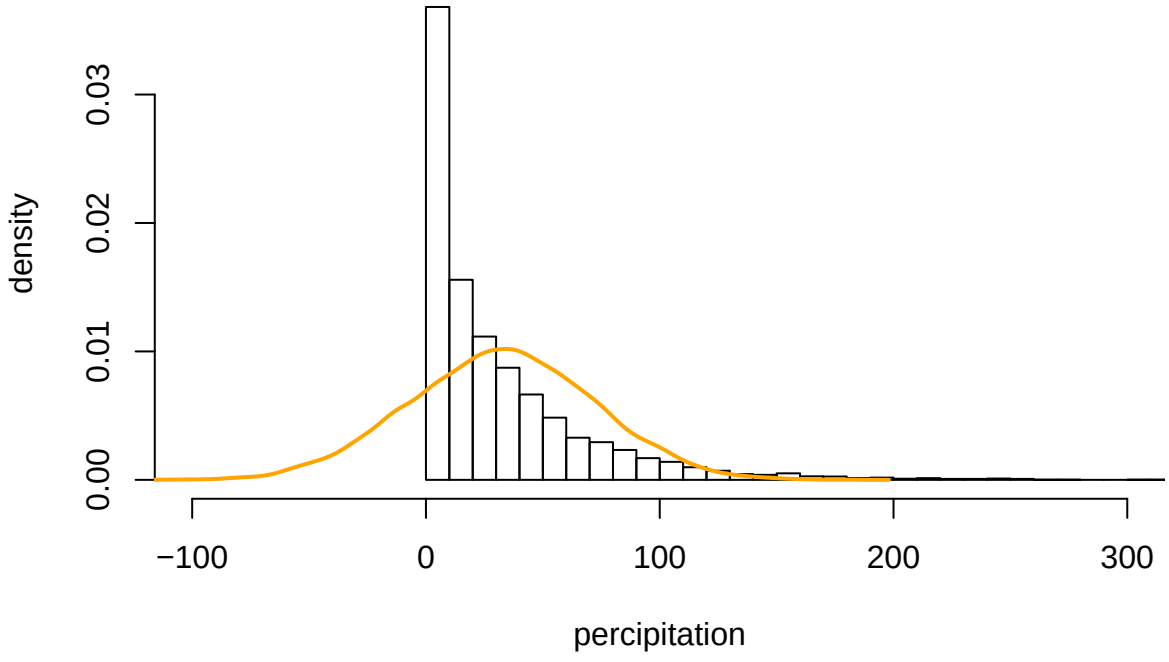
The μ converges to 32.13 which is similar to the data mean, 32.28, and σ^2 converges to 1546.15 which is similar to data variance, 1547.1.

```r
par(mfrow=c(1, 2))
plot(gibbs_params[1:50, "mu"], type = "l", xlab="iteration",
     ylab=expression(mu), main="Convergence")
plot(gibbs_params[1:50, "sigmasq"], type = "l", xlab="iteration",
     ylab=expression(sigma^2), main="Convergence")
```



The convergence happens after just a few iterations.

```
mu <- mean(gibbs_params[, "mu"])
sigmasq <- mean(gibbs_params[, "sigmasq"])
samples <- rnorm(10000, mean=mu, sd=sqrt(sigmasq))

hist(rainfall$V1, freq=FALSE, breaks=50, xlim=c(-100, 300),
     xlab="percipitation", ylab="density", main="")
lines(density(samples), col="orange", lwd=2)
```



The data is obviously not normally distributed so the model is not particularly good which is seen by the orange curve.

b)

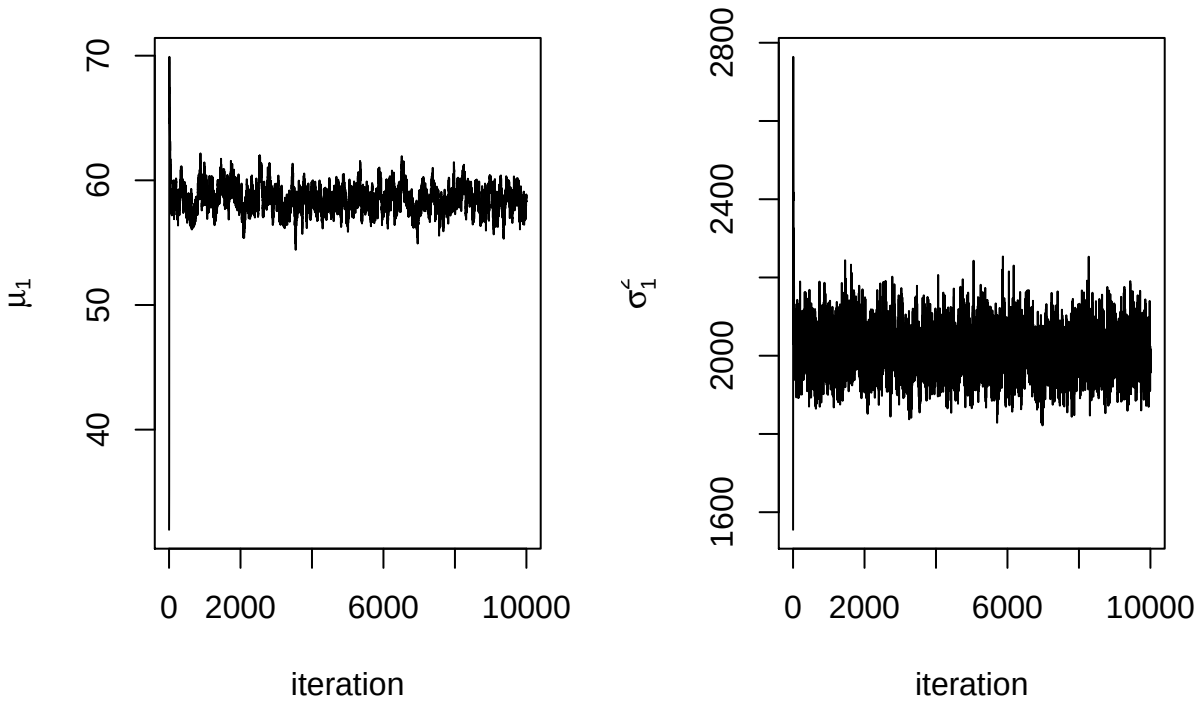
```
nn <- function(I){
    n1 <- sum(I == 1)
    n2 <- sum(I == 2)
    list(n1 = n1,n2 = n2)
}

dd <- function(x, I) {
    d1 <- x[I == 1]
    d2 <- x[I == 2]
    list(d1=d1, d2=d2)
}

pisampler <- function(x, I, hyperparams){
    n <- nn(I)
    rbeta(1, shape1 = n$n1 + hyperparams$a1 , shape2 = n$n2 + hyperparams$a2)
}

sigmasq2sampler <- function(x, I, hyperparams){
    n <- nn(I)
    d <- dd(x, I)

    vnsn1 <-  hyperparams$nu0 * hyperparams$sigmasq0 +
        (n$n1 - 1) * var(d$d1) +
        (((1 / hyperparams$tausq0) * n$n1) / ((1 / hyperparams$tausq0) + n$n1)) *
        ((mean(d$d1) - hyperparams$mu0)^2)

    vnsn2 <-  hyperparams$nu0 * hyperparams$sigmasq0 +
        (n$n2 - 1) * var(d$d2) +
        (((1 / hyperparams$tausq0) * n$n2) / ((1 / hyperparams$tausq0) + n$n2)) *
        ((mean(d$d2) - hyperparams$mu0)^2)

    vn1 <-  hyperparams$nu0 + n$n1
    vn2 <-  hyperparams$nu0 + n$n2

    sn1 <- vnsn1 / vn1
    sn2 <- vnsn2 / vn2

    sigmasq1 <- rinvchisq(1, vn1, sn1)
    sigmasq2 <- rinvchisq(1, vn2, sn2)

    c(sigmasq1, sigmasq2)
}

mu2sampler <- function(x, I, sigmasq, hyperparams){
    d <- dd(x, I)

    sigma1 <- sqrt(taun(d$d1, sigmasq[1], hyperparams))
```

```
    mu1 <- mun(d$d1, sigmasq[1], hyperparams)

    sigma2 <- sqrt(taun(d$d2, sigmasq[2], hyperparams))
    mu2 <- mun(d$d2, sigmasq[2], hyperparams)

    mu1 <- rnorm(1, mu1, sigma1)
    mu2 <- rnorm(1, mu2, sigma2)

    c(mu1, mu2)
}

Isampler <- function(x, pi, mu, sigmasq){
    nom <- (1 - pi) * dnorm(x, mu[2], sqrt(sigmasq[2]))
    denom <- pi * dnorm(x, mu[1], sqrt(sigmasq[1])) + nom
    theta <- nom / denom
    rbinom(length(x), prob = theta, size = 1) + 1 # to get them into 1 and 2
}

ysampler <- function(pi, mu, sigmasq){
    pi * rnorm(1, mu[1], sqrt(sigmasq[1])) +
        (1 - pi) * rnorm(1, mu[2], sqrt(sigmasq[2]))
}

mixedgibbs <- function(x, iter, init, hyperparams){
    params_samples <- matrix(NA, ncol = 5, nrow = iter)
    samples <- rep(NA, iter)
    I <- init

    for (i in 1:iter){
        pi <- pisampler(x, I, hyperparams)
        sigmasq <- sigmasq2sampler(x, I, hyperparams)
        mu <- mu2sampler(x, I, sigmasq, hyperparams)
        I <- Isampler(x, pi, mu, sigmasq)

        samples[i] <- ysampler(pi, mu, sigmasq)
        params_samples[i,] <- c(mu, sigmasq, pi)
    }

    colnames(params_samples) <- c("mu1", "mu2", "sigmasq1", "sigmasq2", "pi")
    list(samples=samples, params=params_samples)
}
```

```
set.seed(123456)

hyperparams <- list(a1=1, a2=1, mu0=1, tausq0=50, nu0=1, sigmasq0=50)
init <- sample(c(1, 2), size=length(rainfall$V1), replace=TRUE)
iter <- 10000

mixed <- mixedgibbs(rainfall$V1, iter, init, hyperparams)
mixed_params <- mixed$params
mixed_samples <- mixed$samples
```

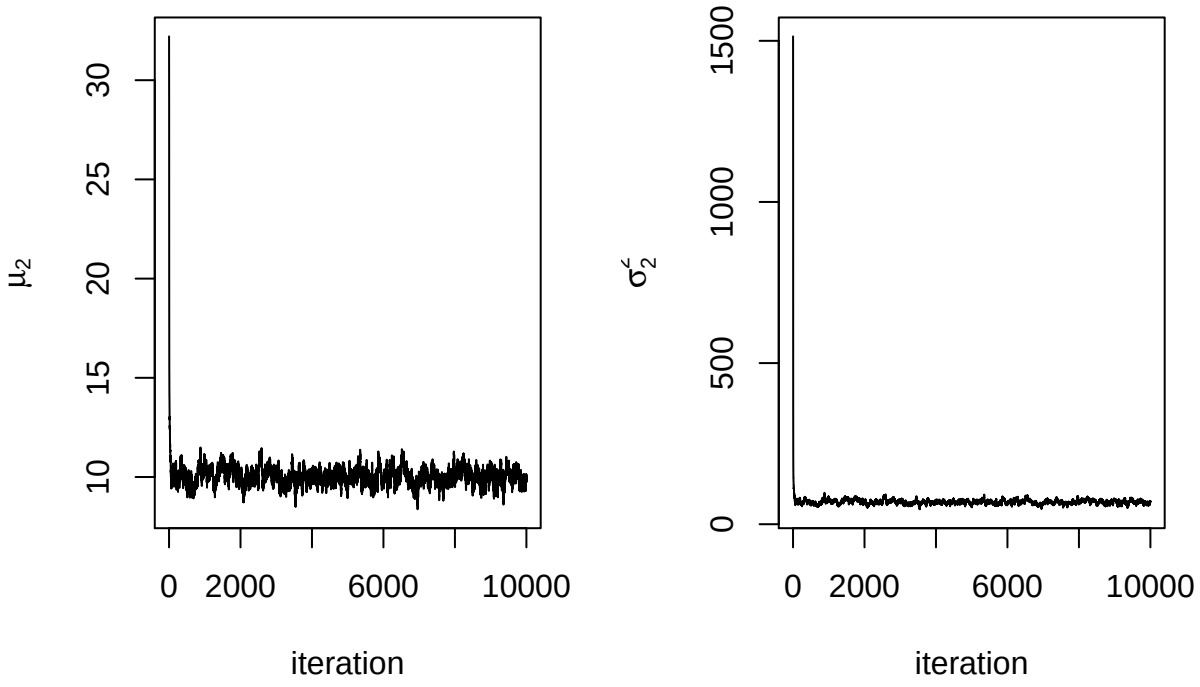
```
par(mfrow = c(1, 2))
plot(x = 1:iter, mixed_params[, 1], type = "l", xlab="iteration", ylab = expression(mu[1]))
```

```
plot(x = 1:iter, mixed_params[, 3], type = "l", xlab="iteration", ylab = expression(sigma[1]^2))
```



The first normal distribution have settled at mean of around 60 and variance of 2000. We can see that it has larger variance in the parameters than the previous model and for the mean it is noticeable autocorrelation between iterations.

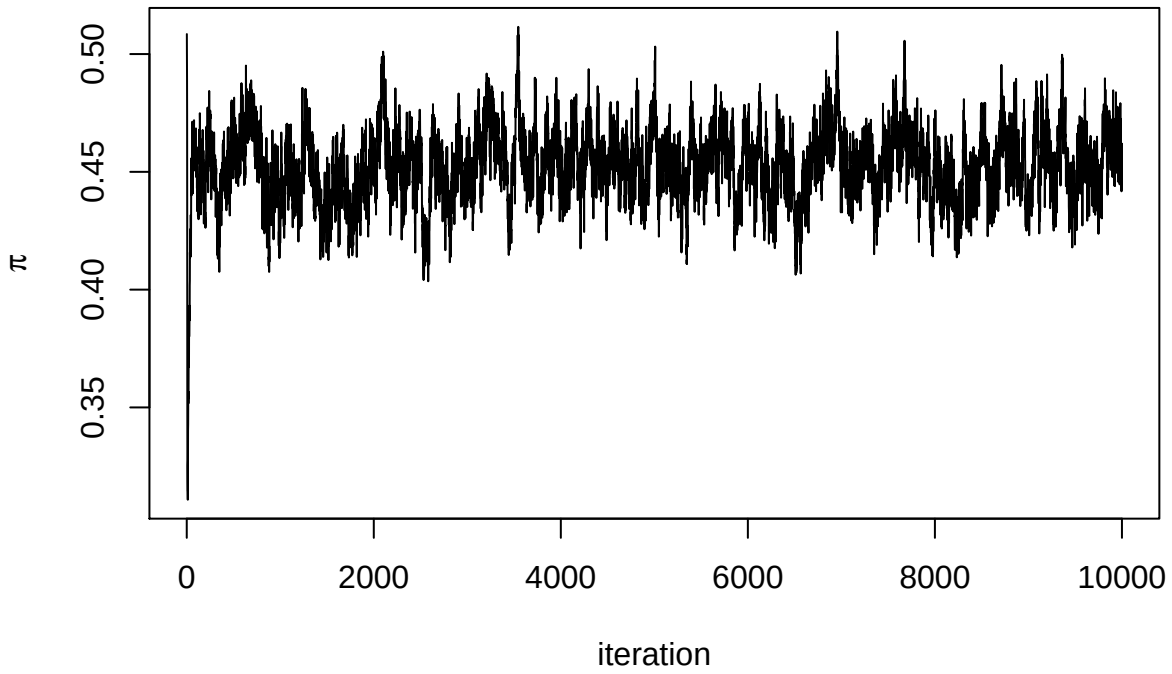
```
par(mfrow = c(1, 2))
plot(x = 1:iter, mixed_params[, 2], type = "l", xlab="iteration", ylab = expression(mu[2]))
plot(x = 1:iter, mixed_params[, 4], type = "l", xlab="iteration", ylab = expression(sigma[2]^2))
```



The second normal distribution have a mean of approximately 10 and variance of around 50. Similarly as

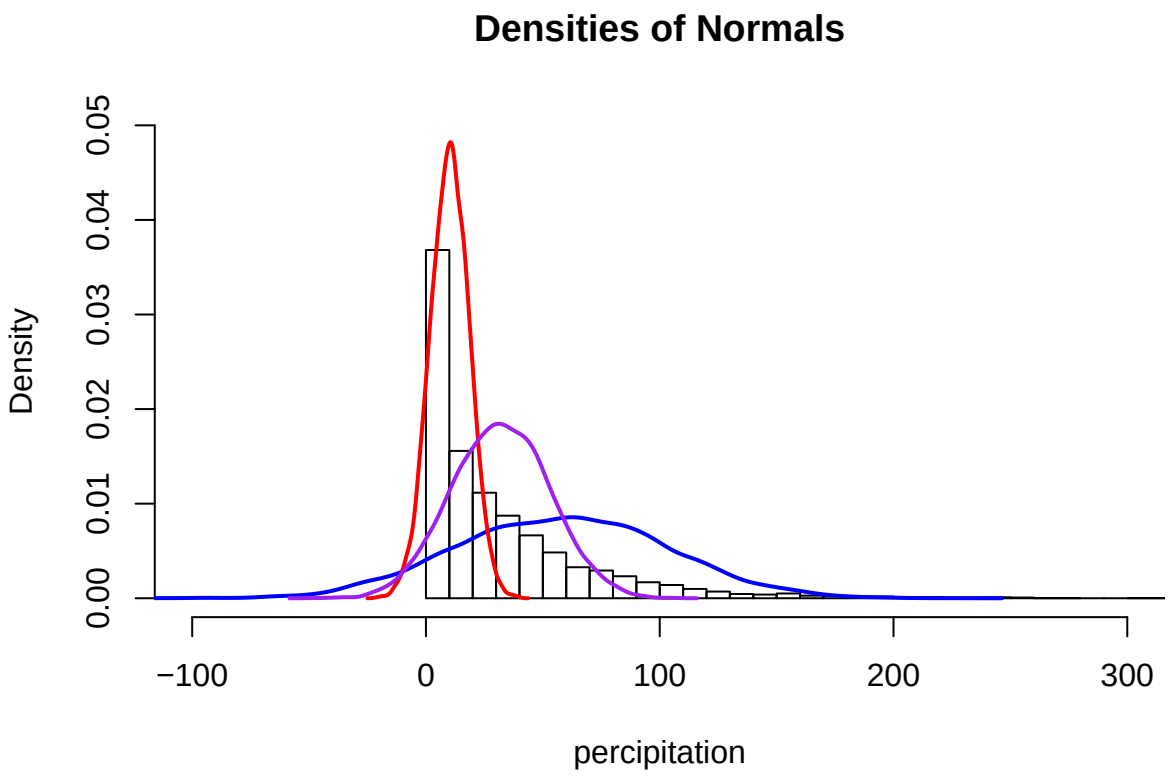
before we can see autocorrelation which indicate that it would probably be beneficial to run the algorithm for longer to get a larger effective sample size that is comparable to the simple normal model in *1a*.

```
par(mfrow = c(1,1))
plot(x = 1:iter, mixed_params[, 5], type = "l", xlab="iteration", ylab = expression(pi))
```



The π is the probability of the observation to be from the first normal distribution and we can see that it is around 45%.

```
n <- 10000

mu1 <- mean(mixed_params[, "mu1"])
sigmasq1 <- mean(mixed_params[, "sigmasq1"])
samples1 <- rnorm(n, mu1, sqrt(sigmasq1))

mu2 <- mean(mixed_params[, "mu2"])
sigmasq2 <- mean(mixed_params[, "sigmasq2"])
samples2 <- rnorm(n, mu2, sqrt(sigmasq2))

pi <- mean(mixed_params[, "pi"])
samples3 <- pi * samples1 + (1 - pi) * samples2


hist(rainfall$V1, freq=FALSE, xlim=c(-100, 300), ylim=c(0, 0.05), breaks=50,
     xlab="percipitation", main="Densities of Normals")
lines(density(samples1), col="blue", lwd=2)
lines(density(samples2), col="red", lwd=2)
lines(density(samples3), col="purple", lwd=2)
```

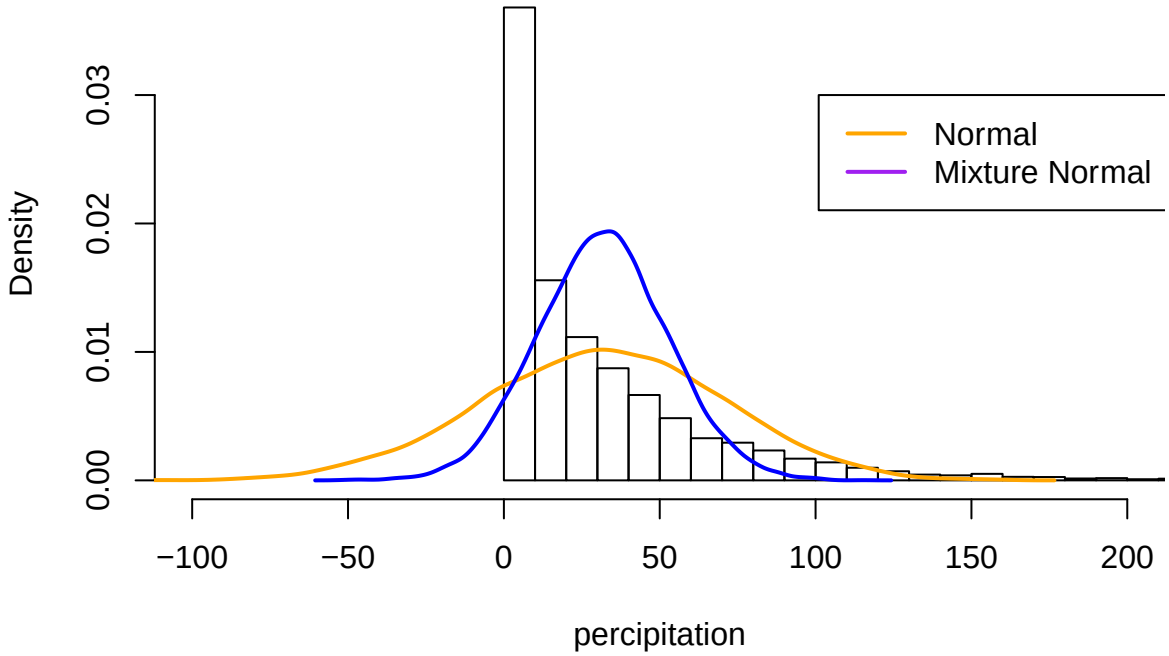


We can see that the two Gaussian distributions in the mixture model represent low values with high density, the red curve, and those with higher values with a wider distribution, the blue curve. Then the model, the purple curve, is an average of those aforementioned distributions. However, as mention before the data is not normally distributed so the final model is far from perfect.

c)

```
n <- 10000
burnins <- 1000 + 1

gibbs_params <- gibbs_params[-(1:burnins),]
mixed_params <- mixed_params[-(1:burnins),]

hist(rainfall$V1, freq=FALSE, breaks=50, xlim=c(-100, 200), main="", xlab="percipitation")

## Model A
mu <- mean(gibbs_params[, "mu"])
sigmasq <- mean(gibbs_params[, "sigmasq"])

samples <- rnorm(n, mu, sqrt(sigmasq))

lines(density(samples), lwd=2, col="orange")

## Model B
mu1 <- mean(mixed_params[, "mu1"])
sigmasq1 <- mean(mixed_params[, "sigmasq1"])

mu2 <- mean(mixed_params[, "mu2"])
```

```r
sigmasq2 <- mean(mixed_params[, "sigmasq2"])

pi <- mean(mixed_params[, "pi"])

samples <- pi * rnorm(n, mu1, sqrt(sigmasq1)) + (1 - pi) * rnorm(n, mu2, sqrt(sigmasq2))

lines(density(samples), lwd=2, col="blue")

legend(101, 0.03, lty=c(1, 1), lwd=c(2, 2), col=c("orange", "purple"),
       legend=c("Normal", "Mixture Normal"))
```



Comparing the normal model and the mixture of normals we conclude that the latter is better in this case but it would probably have been better to choose either a truncated Normal or exponential distribution as the model since we know that percipitation cannot be negative.

Question 2

```
women <- read.table("../data/WomenWork.dat", header = T)
women <- as.matrix(women)
```

a)

```
library(msm)
library(mvtnorm)

betahat <- function(XX, X, y){
  solve(XX) %*% t(X) %*% y
}

mun <- function(XX, betahat, mu0, omega0){
 solve( XX + omega0 ) %*% (XX %*% betahat - omega0 %*% mu0)
}

probitgibbs <- function(X, Y, mu0, omega0, beta0, iter){
  samples <- matrix(ncol = ncol(X), nrow = iter)
  y0 <- Y == 0
  y0n <- sum(y0)
  y1n <- sum(!y0)
  XX <- t(X) %*% X
  omegan <- XX + omega0
  u <- c(rep(1, times = length(Y)))
  beta <- beta0

  for (i in 1:iter){

    xB <- X %*% beta

    u[y0] <- rtnorm(n = y0n,mean = xB[y0], sd = 1, lower = -Inf, upper = 0)
    u[!y0]<- rtnorm(n = y1n,mean = xB[!y0], sd = 1, lower = 0, upper = Inf)

    baehat <- betahat(XX,X,u)
    myn <- mun(XX,baehat,mu0,omega0)

    beta <- as.vector(rmvnorm(1, mean = myn, sigma = 1 * solve(omegan) ))
    samples[i,] <- beta
  }
  samples
}
```

b)

```
betas <- probitgibbs(X = women[,-1], Y = women[,1],
                     mu0 = rep(0, times = ncol(women[,-1])),
                     omega0 = 100* diag(ncol(women[,-1])),
```

```
                    beta = rep(0, times = ncol(women[,-1])), iter = 10000)
probitbetas <- colMeans(betas)
```

c)

```
logprior <- function(beta, mean, sigma){
    dmvnorm(beta, mean = mean, sigma = sigma, log = TRUE)
}

loglikelihood <- function(beta, X, Y){
    linear_prediction <- t(X) %*% beta
    P <- pnorm(linear_prediction, mean = 0, sd = 1)
    probabilities <- Y * log(P) + (1 - Y)*log(1 - P)
    loglike <- sum(probabilities)

    ## if (abs(loglike) == Inf)
    ##     loglike = -20000

    loglike
}

logposterior <- function(beta, X, Y, mean, sigma){
   loglikelihood(beta, X, Y) + logprior(beta, mean, sigma)
}

tau <- 10
mu <- rep(0,8)
sigma <- tau^2 * diag(8)

womenX <- as.matrix(women[,2:ncol(women)])
womenY <- as.matrix(women[,1])

quadapprox <- optim(par = matrix(rep(0, 8), ncol = 1),
                fn = logposterior, method = "BFGS", hessian = TRUE,
                X = t(womenX), Y = womenY,
                mean = mu, sigma = sigma,
                control=list(fnscale=-1))

mu <- quadapprox$par
sigma <- -solve(quadapprox$hessian)
quadbetas <- rmvnorm(nrow(betas), mean=mu, sigma=sigma)
```

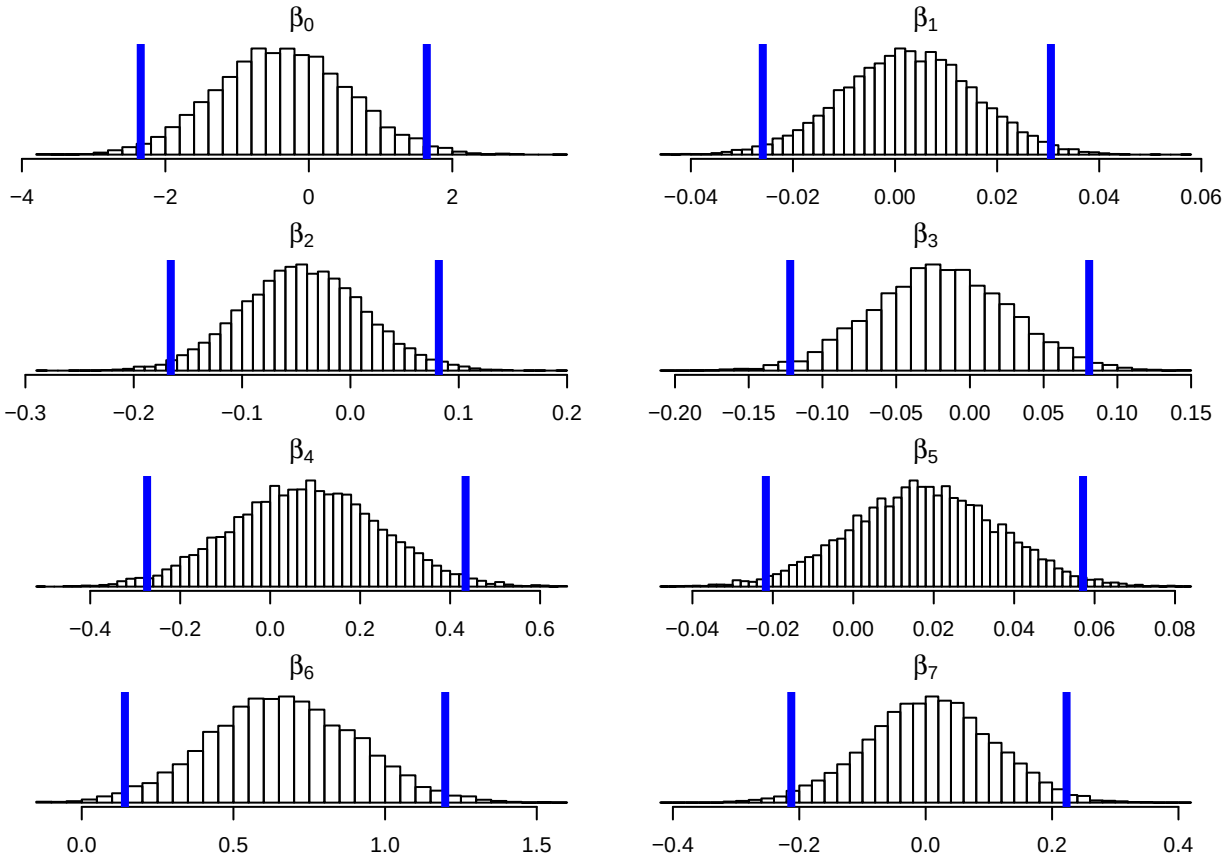
```
options(digits=3)

probitbetas
```

```
## [1] -0.00572 -0.00944  0.06510  0.08245 -0.00918 -0.03144 -0.15049 -0.00455
```

```
as.vector(quadapprox$par)
```

```
## [1]  0.3845 -0.0121  0.1084  0.1022 -0.0902 -0.0496 -0.8157 -0.0124
```

```
diff <- betas - quadbetas

par(mfrow=c(4, 2), mar=c(2, 1, 2, 1))
for (i in 1:ncol(betas)) {
    ci <- quantile(diff[, i], c(0.015, 0.985))
    hist(diff[, i], xlab="", ylab="", main=bquote(beta[.(i-1)]),
         breaks=50, yaxt="n")
    abline(v=ci[1], lwd=4, col="blue")
    abline(v=ci[2], lwd=4, col="blue")
}
```



The histograms above show the distributions for the difference, $\beta_i^{\text{Gibbs}} - \beta_i^{\text{Norm}}$ for $i = 0, \ldots, 7$, and the blue lines are the 97% credibility intervals. We can see that only in the β_6 parameter that 0 is not within the credibility interval which indicate that there is a difference in that parameter. However, the rest of the parameters do not differ statistically so we conclude that the quadratic (normal) approximation is a decent approximation in this case.

Lab 4

Question 1

```
bid <- read.table("../data/eBayNumberOfBidderData.dat",header = TRUE)
```

a)

```
poiglm <- glm(nBids ~ . - 1, data = bid, family = poisson(link = "log"))
summary(poiglm)
```

```
##
## Call:
## glm(formula = nBids ~ . - 1, family = poisson(link = "log"),
##     data = bid)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.580  -0.722  -0.044   0.527   2.461
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Const         1.0724     0.0308   34.85  < 2e-16 ***
## PowerSeller  -0.0205     0.0368   -0.56    0.577
## VerifyID     -0.3945     0.0924   -4.27  2.0e-05 ***
## Sealed        0.4438     0.0506    8.78  < 2e-16 ***
## Minblem      -0.0522     0.0602   -0.87    0.386
## MajBlem      -0.2209     0.0914   -2.42    0.016 *
## LargNeg       0.0707     0.0563    1.25    0.210
## LogBook      -0.1207     0.0290   -4.17  3.1e-05 ***
## MinBidShare  -1.8941     0.0712  -26.59  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 6264.01  on 1000  degrees of freedom
## Residual deviance:  867.47  on  991  degrees of freedom
## AIC: 3610
##
## Number of Fisher Scoring iterations: 5
```

b)

```
library(mvtnorm)

logprior <- function(beta, mu, sigma){
    dmvnorm(beta, mean = mu, sigma = sigma, log = TRUE)
}
```

```r
loglikelihood <- function(beta, X, Y){
    linear_prediction <- t(X) %*% beta
    probabilities <- Y * linear_prediction - exp(linear_prediction)
    loglike <- sum(probabilities)

    ## if (abs(loglike) == Inf)
    ##     loglike = -20000

    loglike
}

## loglikelihood <- function(beta, X, Y) {
##   linear_prediction <- t(X) %*% beta
##   probs <- dpois(Y , lambda = exp(linear_prediction), log = TRUE)
##   sum(probs)
## }

logposterior <- function(beta, X, Y, prior_mu, prior_sigma){
    loglikelihood(beta, X, Y) + logprior(beta, prior_mu, prior_sigma)
}

X <- as.matrix(bid[,-1])
Y <- as.matrix(bid[,1])

mu <- rep(0, ncol(X))
sigma <- 100 * solve(t(X) %*% X)

optpost <- optim(par = matrix(rep(0, ncol(X)), ncol = 1),
                 fn = logposterior, method = "BFGS", hessian = TRUE,
                 X = t(X), Y = Y,
                 prior_mu= mu, prior_sigma = sigma,
                 control=list(fnscale=-1))
hessian <- optpost$hessian
```

```r
as.vector(optpost$par)
```

```
## [1]  1.0698 -0.0205 -0.3930  0.4436 -0.0525 -0.2212  0.0707 -0.1202 -1.8920
```

```r
as.vector(coef(poiglm))
```

```
## [1]  1.0724 -0.0205 -0.3945  0.4438 -0.0522 -0.2209  0.0707 -0.1207 -1.8941
```

c)

```r
targetdensity <- function(theta, prior_mu, prior_sigma, X, Y, ...) {
    likelihood <- dpois(Y, lambda = exp(t(X) %*% t(theta)), log = TRUE)
    prior <- dmvnorm(theta, mean = prior_mu, sigma = prior_sigma, log = TRUE)
    sum(likelihood) + prior
}

proposaldensity <- function(x, mu, prop_sigma, ...){
    dmvnorm(x, mean = mu, sigma = prop_sigma, log = TRUE)
}
```

```r
proposalsampler <- function(mu, prop_sigma, ...){
    matrix(rmvnorm(1, mean = mu, sigma = prop_sigma), nrow = 1)
}

metropolis_hastings <- function(prop_sampler, log_prop_func, log_targ_post_func, X0, iters, ...){
    x <- X0
    values <- matrix(0, ncol = length(X0), nrow = iters + 1)
    values[1,] <- X0

    alpha <- function(x, y, ...) {
        numerator <- log_targ_post_func(y, ...) + log_prop_func(x, y, ...)
        denominator <- log_targ_post_func(x, ...) + log_prop_func(y, x, ...)
        exp(numerator - denominator)
    }

    for (i in 1:iters) {
        y <- prop_sampler(x, ...)
        u <- runif(1)

        if (u < alpha(x, y, ...)) {
            x <- y
        }

        values[i+1,] <- x
    }

    values
}

iters <- 5000
X0 <- rep(0, times = ncol(X))

params <- list(
    prop_sampler = proposalsampler,
    log_prop_func = proposaldensity,
    log_targ_post_func = targetdensity,
    X0 = matrix(rep(0, times = ncol(X)), nrow = 1),
    iters = iters,
    X = t(X),
    Y = Y,
    prior_mu = rep(0, times = ncol(X)),
    prior_sigma = 100 * solve(t(X) %*% X),
    prop_sigma = 0.6 * -solve(hessian)
)

res <- do.call(metropolis_hastings, params)

colMeans(res)
```

```
## [1]  1.0347 -0.0115 -0.3843  0.4311 -0.0506 -0.2077  0.0715 -0.1140 -1.8346
```

d

```
Xpred <- matrix(c(1, 1, 1, 1, 0, 0, 0, 1, 0.5), nrow = 1)
predsmaples <- rpois(10000, lambda = exp(Xpred %*% t(res)))

mean(predsmaples == 0)
```

```
## [1] 0.349
```

```
hist(predsmaples, breaks = 50)
```