Week 6: Oct 3 - Oct 7

Now can we use the full conditional distributions to draw samples from the joint posterior?

Suppose we had $\sigma^{2(1)}$, a single sample from the marginal posterior distribution $p(\sigma^2|y_1, \ldots, y_n)$. Then we could sample:

and $\{\theta^{(1)}, \sigma^{2(1)}\}$ would be a sample from the joint posterior distribution $p(\theta, \sigma^2|y_1, \ldots, y_n)$. Now using $\theta^(1)$ we can generate another sample of σ^2 from

This sample $\{\theta^{(1)}, \sigma^{2(2)}\}$ would also be a sample from the joint posterior distribution. This process follows iteratively. However, we don't actually have $\sigma^{2(1)}$.

Gibbs Sampler

The distributions $p(\theta|y_1, \ldots, y_n, \sigma^2)$ and $p(\sigma^2|y_1, \ldots, y_n, \theta)$ are known as the full conditional distributions, that is they condition on all other values and parameters. The Gibbs sampler uses these full conditional distributions and the procedure follows as:

0.

1.

2.

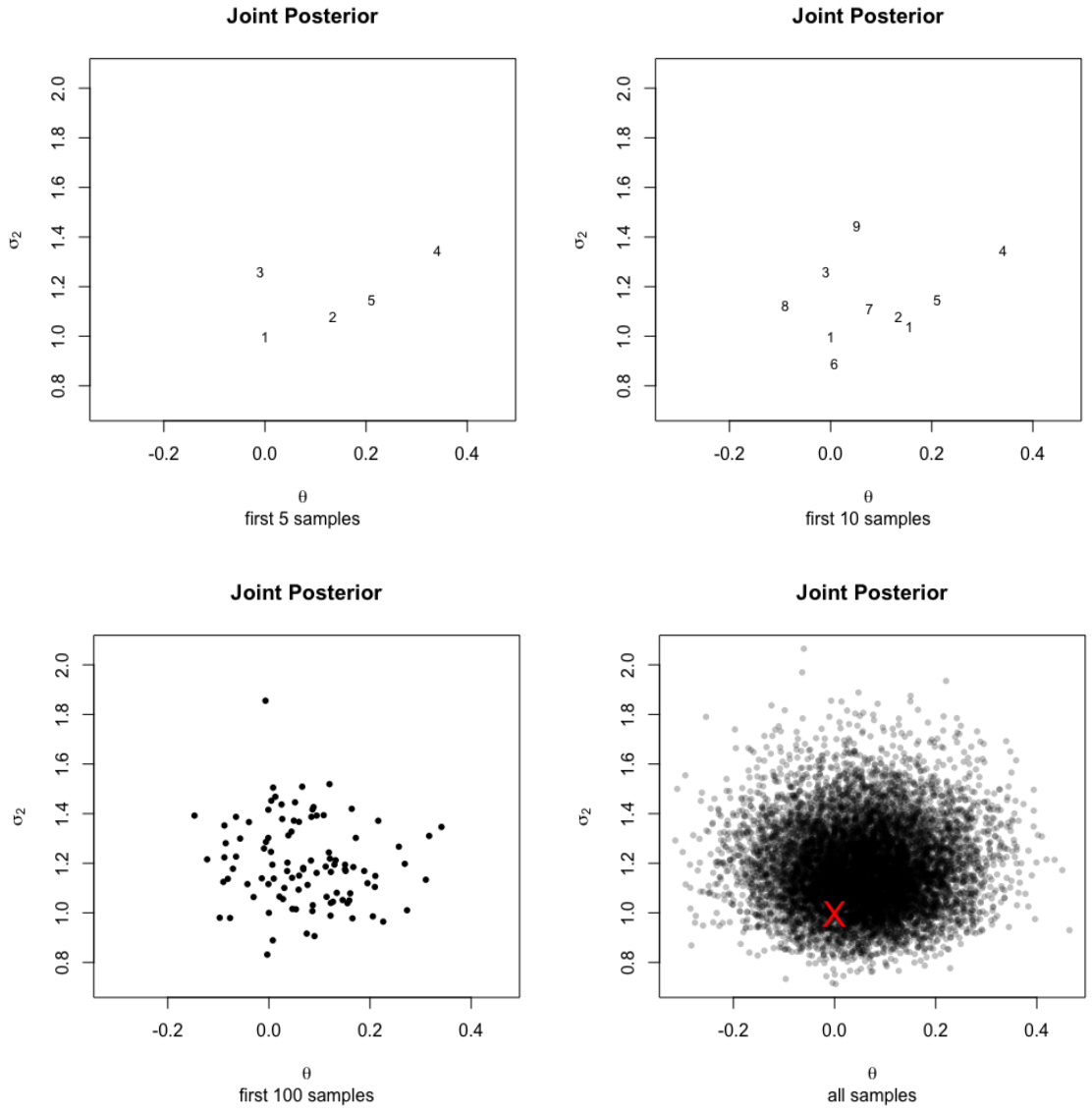
3.

The code and R output for this follows.

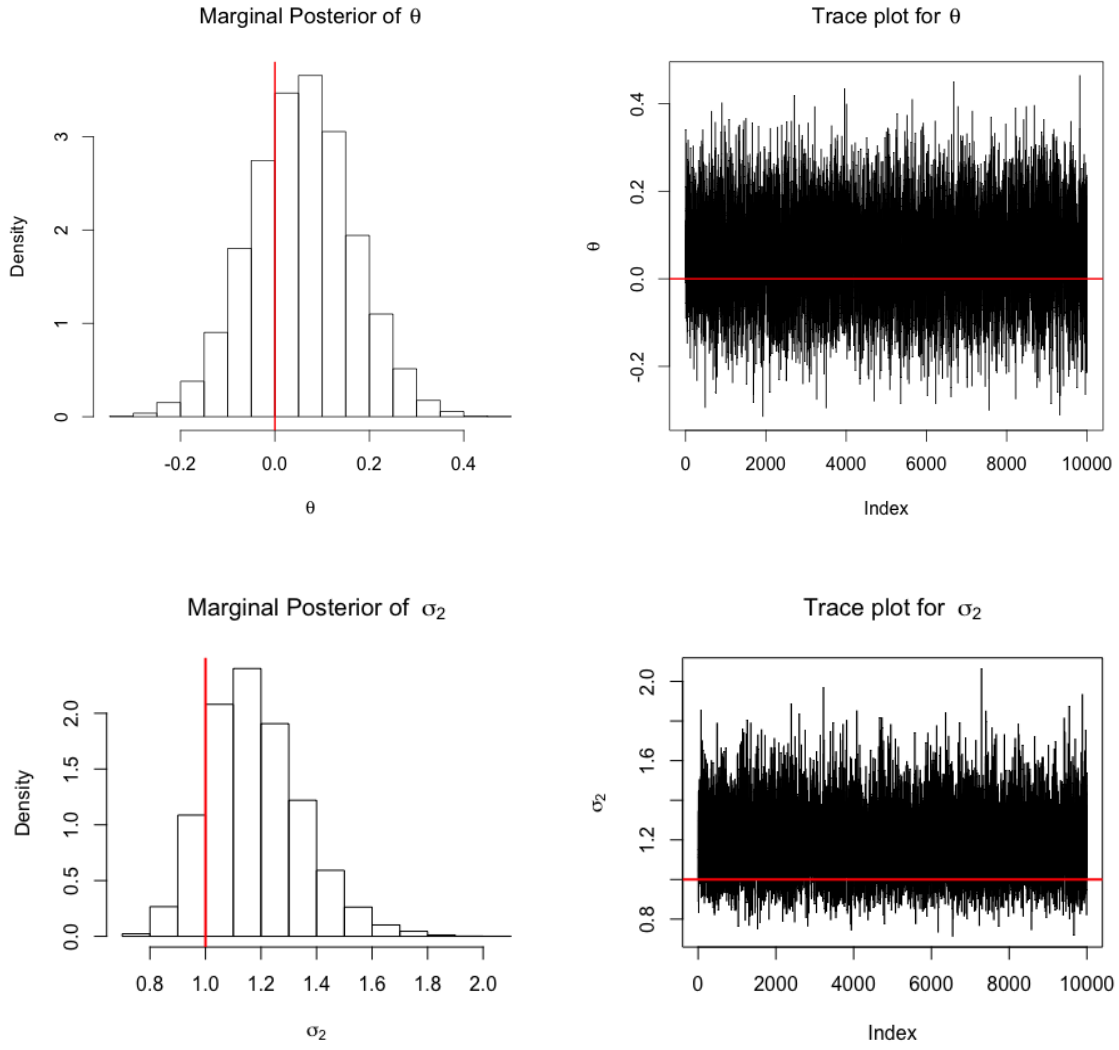
```
######### First Gibbs Sampler
set.seed(09222016)
### simulate data
num.obs <- 100
mu.true <- 0
sigmasq.true <- 1
y <- rnorm(num.obs,mu.true,sigmasq.true)
mean.y <- mean(y)
var.y <- var(y)
```

```
### initialize vectors and set starting values and priors
num.sims <- 10000
Phi <- matrix(0,nrow=num.sims,ncol=2)
Phi[1,1] <- 0 # initialize theta
Phi[1,2] <- 1 # initialize (1/sigmasq)
mu.0 <- 0
tausq.0 <- 1
nu.0 <- 1
sigmasq.0 <- 1

for (i in 2:num.sims){
  # sample theta from full conditional
  mu.n <- (mu.0 / tausq.0 + num.obs * mean.y *Phi[(i-1),2]) / (1 / tausq.0 + num.obs * Phi[(i-1),2] )
  tausq.n <- 1 / (1/tausq.0 + num.obs * Phi[(i-1),2])
  Phi[i,1] <- rnorm(1,mu.n,sqrt(tausq.n))

  # sample (1/sigma.sq) from full conditional
  nu.n <- nu.0 + num.obs
  sigmasq.n.theta <- 1/nu.n*(nu.0*sigmasq.0 + sum((y - Phi[i,1])^2))
  Phi[i,2] <- rgamma(1,nu.n/2,nu.n*sigmasq.n.theta/2)
}

# plot joint posterior
plot(Phi[1:5,1],1/Phi[1:5,2],xlim=range(Phi[,1]),ylim=range(1/Phi[,2]),pch=c('1','2','3','4','5'),cex=.8,
     ylab=expression(sigma[2]), xlab = expression(theta), main='Joint Posterior',sub='first 5 samples')

plot(Phi[1:10,1],1/Phi[1:10,2],xlim=range(Phi[,1]),ylim=range(1/Phi[,2]),pch=as.character(1:15),cex=.8,
     ylab=expression(sigma[2]), xlab = expression(theta), main='Joint Posterior',sub='first 10 samples')

plot(Phi[1:100,1],1/Phi[1:100,2],xlim=range(Phi[,1]),ylim=range(1/Phi[,2]),pch=16,col=rgb(0,0,0,1),cex=.8,
     ylab=expression(sigma[2]), xlab = expression(theta), main='Joint Posterior',sub='first 100 samples')

plot(Phi[,1],1/Phi[,2],xlim=range(Phi[,1]),ylim=range(1/Phi[,2]),pch=16,col=rgb(0,0,0,.25),cex=.8,
     ylab=expression(sigma[2]), xlab = expression(theta), main='Joint Posterior',sub='all samples')
points(0,1,pch='X',col='red',cex=2)

# plot marginal posterior of theta
hist(Phi[,1],xlab=expression(theta),main=expression('Marginal Posterior of ' ~ theta),probability=T)
abline(v=0,col='red',lwd=2)
# plot marginal posterior of sigmasq
hist(1/Phi[,2],xlab=expression(sigma[2]),main=expression('Marginal Posterior of ' ~ sigma[2]),probability=T)
abline(v=1,col='red',lwd=2)

# plot trace plots
plot(Phi[,1],type='l',ylab=expression(theta), main=expression('Trace plot for ' ~ theta))
abline(h=0,lwd=2,col='red')
plot(1/Phi[,2],type='l',ylab=expression(sigma[2]), main=expression('Trace plot for ' ~ sigma[2]))
abline(h=1,lwd=2,col='red')

# compute posterior mean and quantiles
c(mean(Phi[,1]),mean(1/Phi[,2]))
quantile(Phi[,1],probs=c(.025,.975))
quantile(1/Phi[,2],probs=c(.025,.975))
```

So what do we do about the starting point? We will see that given a reasonable starting point the algorithm



will converge to the true posterior distribution. Hence the first (few) iterations are regarded as the burn-in period and are discarded (as they have not yet reached the true posterior).



More on the Gibbs Sampler

The algorithm previously detailed is called the *Gibbs Sampler* and generates a dependent sequence of parameters $\{\phi_1, \phi_2, \ldots, \phi_n\}$. This is in contrast to the Monte Carlo procedure we previously detailed, including the situation where $p(\theta|\sigma^2) \sim N(\mu_0, \sigma^2/\kappa_0)$.

The Gibbs Sampler is a basic Markov Chain Monte Carlo (MCMC) algorithm. A Markov chain is a stochastic process where the current state only depends on the previous state. Formally

Depending on the class interests, we may return to talk more about the theory of MCMC later in the course, but

the basic ideas are:

That is the sampling distribution of the draws from the MCMC algorithm approach the desired target distribution (generally a posterior in Bayesian statistics) as the number of samples j goes to infinity. The is not dependent on the starting values of $\phi^{(0)}$, but poor starting values will take longer for convergence. Note this will be more problematic when we consider another MCMC algorithm, the Metropolis-Hastings sampler. Given the equation above, for most functions $g(.)$:

Thus we can approximate expectations of functions of ϕ using the sample average from the MCMC draws, similar to our Monte Carlo procedures presented earlier.

Estimation vs. Approximation

There are a few elements of a Bayesian data analysis:

1. *Model Specification:*

2. *Prior Specification:*

3. *Posterior Summary:*

For most models we have discussed thus far, $p(\phi|y_1, \ldots, y_n)$ is known in closed form or easy to sample from using Monte Carlo procedures. However, in more sophisticated settings, $p(\phi|y_1, \ldots, y_n)$ is complicated, and

hard to write down or sample from. In these cases, we study $p(\phi|y_1, \ldots, y_n)$ by looking at MCMC samples. Thus, Monte Carlo and MCMC sampling algorithms:

- are not models,
- they do not generate "more information" than is in $y_1, \ldots, y_n$ and $p(\phi)$
- they are simply 'ways of looking at' $p(\phi|y_1, \ldots, y_n)$

For example, if we have Monte Carlo samples $\phi^{(1)}, \ldots, \phi^{(J)}$ that are approximate draws from $p(\theta|y_1, dots, y_n)$, then these sample help describe $p(\phi|y_1, \ldots, y_n)$, for example:

-

-

To keep the distinction between *estimation* and *approximation* clear, commonly *estimation* is used do describe how we use $p(\phi|y_1, \ldots, y_n)$ to make inferences about ϕ and use *approximation* to describe the use of Monte Carlo (including MCMC) procedures to approximate integrals.

MCMC Exercise

The purpose of Monte Carlo or MCMC algorithms is to generate a sequence of sample draws $\{\phi^{(1)}, \ldots, \phi^{(J)}\}$ such that we can approximate

$$\approx \int g(\phi)p(\phi)d\phi,$$

where $g(.)$ could be an expectation, quantile, or other property of the posterior distribution. In order for this approximation to be 'good' the empirical distribution of the simulated sequence $\{\phi^{(1)}, \ldots, \phi^{(J)}\}$ needs to lok like the target distribution $p(\phi)$. Note the target distribution is typically a posterior, but this notation allows $p(\phi)$ to be a generic distribution function.

Monte Carlo procedures allow us to generate independent sample from the target distribution; hence, these samples are representative of the target distribution. The only question is how many sample to we need to attain

a specified level of precision (usually CLT theory for approximating expectations). However, with MCMC samples we generate dependent samples and all we are guaranteed theoretically is:

$$= \int_A p(\phi) d\phi$$

In other words, our posterior samples from the MCMC algorithm will *eventually* converge. However, there is no guarantee that the convergence can even be acheived in a practical amount of computing time.

Example. An example that is notoriously difficult to acheive convergence is high-dimensional multimodal settings. Similar to optimization algorithms, MCMC samplers can 'get stuck' in a single mode.

Exercise. Will be given as an in class lab/quiz Consider the mixture distribution described on p. 99 (Hoff). This distribution is a joint probability distribution of a discrete variable $\delta = \{1, 2, 3\}$, denoting which mixture component the mass comes from and a continuous variable θ. The target density is $\{Pr(\delta = 1), Pr(\delta = 2), Pr(\delta = 3)\} = (.45, .10, .45)$ and $p(\theta|\delta = i) \sim N(\theta; \mu_i, \sigma_i^2)$ where $\{\mu_1, \mu_2, \mu_3\} = (-3, 0, 3)$ and $\sigma_i^2 = 1/3$ for $i \in \{1, 2, 3\}$.

1. Generate 1000 samples of θ from this distribution using a Monte Carlo procedure. Hint: first generate $\delta^{(i)}$ from the marginal distribution $p(\delta)$ and then generate $\theta^{(i)}$ from $p(\theta|\delta)$ Plot your samples in a histogram form and superimpose a curve of the density function. Comment on your samples, do they closely match the true distribution?

2. Next, generate samples from a Gibbs sampler using the full conditional distributions of θ and δ. You already know the form of the full conditional for θ from above. The full conditional distribution for δ is given below:

$$Pr(\delta = d|\theta) = \frac{Pr(\delta = d) \times p(\theta|\delta = d)}{\sum_{d=1}^{3} Pr(\delta = d) \times p(\theta|\delta = d)}$$

   Hint: for $p(\theta|\delta = d)$ evaluate θ from a normal distribution with parameters $\{\mu_d, \sigma_d^2\}$. Intialize θ at 0.

   (a) Generate 100 samples using this procedure. Plot your samples as a histogram with the true density superimposed on the plot. Also include a plot of your θ value on the y-axis and the iteration number on the x-axis. This is called a trace plot, and allows your to visualize the movement of your MCMC *particle*. Comment on how close your samples match the true density. What does the trace plot reveal about the position of θ over time (the iterations)? Does the proportion of the time the sample spends in each state (δ) match the true probabilities?

   (b) Repeat for 1000 samples.

   (c) Repeat for 10000 samples.

3. Now repeat part 2, but instead initialize θ at 100. How does this change the results from part 2? Looking at trace plots, do the chains from the three plots seem to be similar?

   When complete, turn your code in as a R markdown document for quiz 3.

MCMC Diagnostics

A useful way to think about an MCMC sampler is that there is a *particle* moving through and exploring the parameter space. For each region, or set, A the particle needs to spend time proportional to the target probability, $\int_A p(\phi)d\phi$ Consider the three modes from the exercise conducted in class and denote these three modes as A_1, A_2, A_3. Given the weights on the mixture components the particle should spend more time in A_1 and A_3 than A_2. However, if the particle was initialized in A_2 we'd hope that the number of iterations are large enough that:

1.

2.

The technical term associated with item 1 is *stationarity*, which means the chain has conveged to the target distribution. For the models we have seen thus far, convergence happens quite rapidly, but we will look at this in more depth later on. The second item is focused on the speed the particle moves through the target distribution, this is referred to as *mixing*. An independent sampler like the Monte Carlo procedures we have seen have perfect mixing as each sample is independently drawn from the target distribution. The MCMC samples can be highly correlated and tend to get stuck in certain regions of the space.

People often quantify mixing properties of MCMC samples using the idea of effective sample size. To understand this, first consider the variance of independent Monte Carlo samples:

$$Var_{MC}[\bar{\phi}] =$$

where $\bar{\phi} = \sum_{j=1}^{J} \phi^{(j)}/J$. The Monte Carlo variance is controlled by the number of samples obtained from the algorithm. In a MCMC setting, consecutive samples $\phi^{(j)}$ and $\phi^{(j+1)}$ are not independent, rather they are usually positively correlated. Once stationarity has been acheived, the variance of the MCMC algorithm can be

expressed as:

$$Var_{MCMC}[\phi] = \cdots = Var_{MC}[\bar{\phi}] +$$

where ϕ_0 is the true value of the integral, typically $E[\phi]$. Now if two consecutive samples are highly correlated the variance of the estimator will be much larger than that of an Monte Carlo procedure with the same number of iterations. This is captured in the idea of the **effective sample**. The effective sample size is computed such that:

$$Var_{MCMC}[\bar{\phi}] = \frac{Var[\phi]}{S_{eff}},$$

where S_{eff} can be interpreted as the number of independent Monte Carlo samples necessary to give the same precision as the MCMC samples. Note that the R function `effectiveSize` in the "coda" package will calculate the effective sample size of MCMC output.

We will talk more about MCMC diagnostics after introducing the Metropolis-Hastings algorithm later in class, but the general procedure is:

1.

2.

An easy solution, especially in the context of Gibbs Sampling is to look at trace plots and histograms of marginal posterior distributions. In conjunction with ESS (Effective Sample Size) calculations this usually gives a good sense of convergence. In other situations, combining visuals displays (trace plots) with other statisics Gelman's R statistic or QDE is a good strategy.

The big picture idea with MCMC, is that we want to guarantee that our algorithm has:

1.

2.

Posterior Model Checks on Normal Model

Exercise. Consider a similar scenario to the code for the first Gibbs sampler. Again 100 data points have been generated.

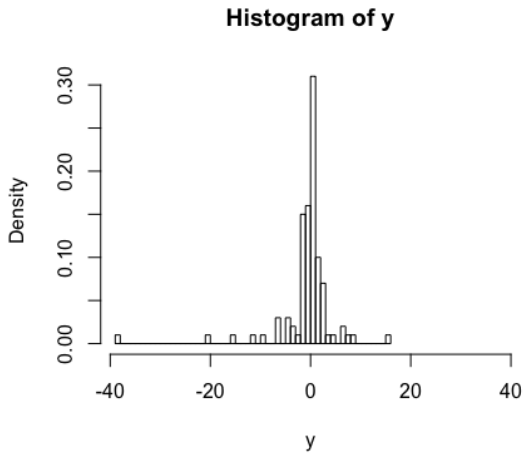
1. A histogram of the data is shown later as figure (a). What are your thoughts about this data?

2. Now assume you used your MCMC code and came up with figures (b) - (e). Comment on the convergence and the marginal posterior distributions.

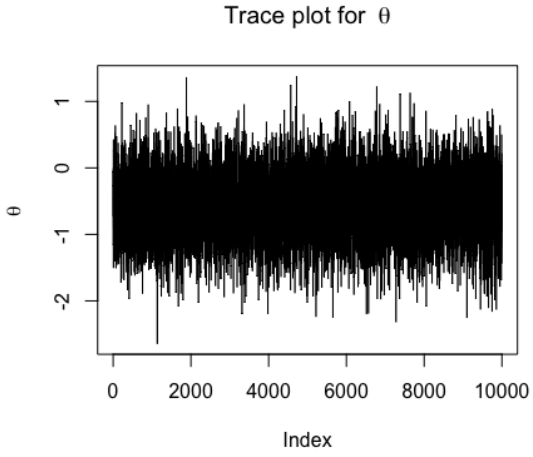
3. As a final check you decide to use your MCMC samples to compute the posterior predictive distribution, $p(y^*|y_1, \ldots, y_n)$. Computationally this can be acheived by using each pair $\{\theta^{(i)}, \sigma^{2(i)}\}$ and then simulating from $N(y^*; \theta^{(i)}, \sigma^{2(i)})$. In R this can be done with one line of code:

   ```
   post.pred <- rnorm(num.sims,mean=Phi[,1],sd = sqrt(1/Phi[,2]))
   ```

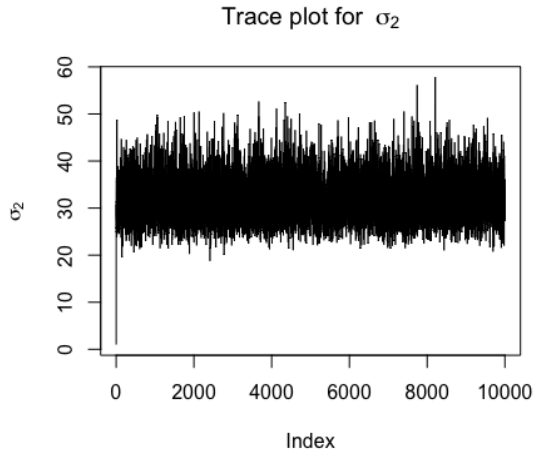
   Now compare your posterior predictive distribution with the observed data. Are you satisfied that the posterior predictive distribution represents the actual data? Why or why not?



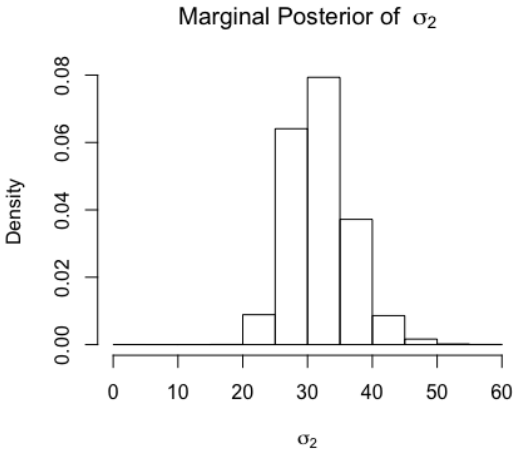
(a) Histogram of the Data



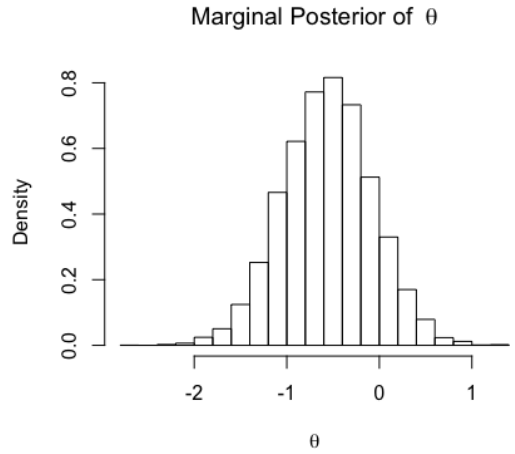
(b) Trace plot for θ

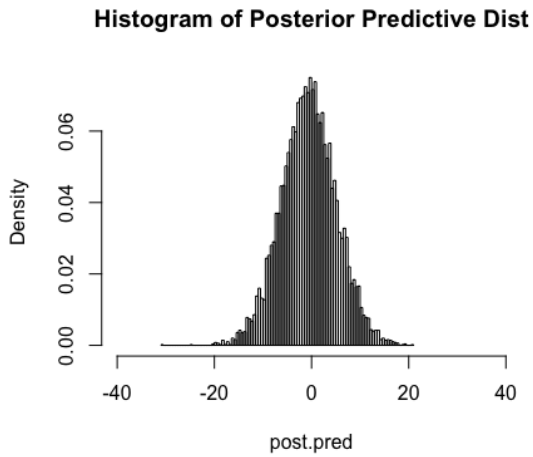


(c) Trace plot for σ^2



(d) Histogram of marginal posterior for σ^2





(e) Histogram of marginal posterior for θ