Homework 3

Michael Pena

2024-10-27

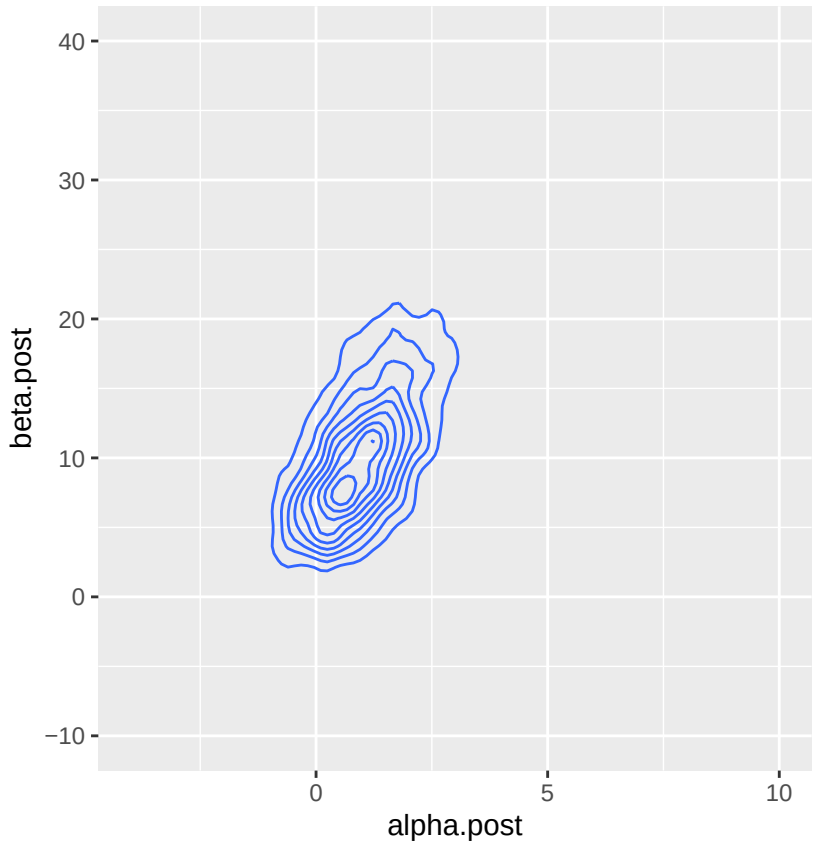
Book Question 3.11

####(a)

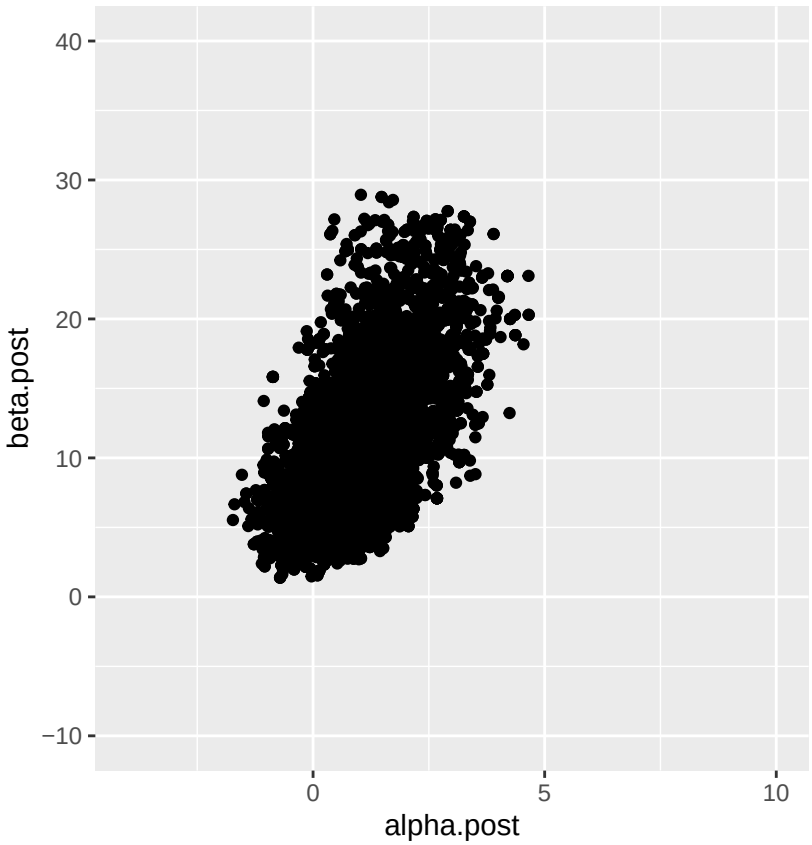
```
# Bioassay
J = 4
n = c(5,5,5,5)
y = c(0, 1, 3, 5)
x = c(-0.86, -0.30, -0.05, 0.73)

# Y_i|theta_i ~ Bin(n_i, theta_i); logit(theta_i) = alpha + beta*x_i
# p(alpha, beta) = 1

Prior = function(alpha, beta){
  dmvnorm(x = c(alpha,beta), mean = c(0,10), sigma = matrix(c(4,10,10,100),2,2))
}

Likelihood = function(Y, N, X, alpha, beta){
  th = inv.logit(alpha + beta*X)
  prod(dbinom(Y, N, th))
}

rProposal = function(n, alpha, beta, s, S){
  rmvnorm(1, mean = c(alpha, beta), sigma = s*S)
}

### initialize
accept = 0
s = 1 # tuning param.
S = matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2)

# S = matrix(c(1, 2.3, 2.3, 18), nrow = 2, ncol = 2)
###

temp = y/n # "MLE's of thetas"
temp = c(0.01, 0.2, 0.6, 0.9) # fixing the boundries
ifit = summary(lm(logit(temp) ~ x)) # est. are reasonable est. for alpha and beta

alpha.post = beta.post = numeric()
alpha.post[1] = -0.3356453
beta.post[1] = 4.2419651

B = 8000
```

```r
for(b in 2:(B+1)){

  #Joint MH step: alpha, beta|data
  ab.star = rProposal(1, alpha.post[(b-1)], beta.post[(b-1)], s, S)

  # Ratio of Densities
  r.num = Likelihood(y, n, x, ab.star[1], ab.star[2]) *
          Prior(ab.star[1], ab.star[2])

  r.den = Likelihood(y, n, x, alpha.post[(b - 1)], beta.post[(b - 1)]) *
          Prior(alpha.post[(b - 1)], beta.post[(b - 1)])

  r = r.num / r.den

  # Accept/Reject
  if(runif(1) < min(1, r)){
    alpha.post[b] = ab.star[1]
    beta.post[b] = ab.star[2]
    accept = accept + 1
  }else{
    alpha.post[b] = alpha.post[(b-1)]
    beta.post[b] = beta.post[(b-1)]
  }

}

# plot(alpha.post, type = "l")
# plot(beta.post, type = "l")
# burn = 500
# acf(alpha.post[-c(1:burn)])
# acf(beta.post[-c(1:burn)])
#
# var(alpha.post[-c(1:burn)])
# cov(alpha.post[-c(1:burn)], beta.post[-c(1:burn)])
# var(beta.post[-c(1:burn)])
```

```r
# contour and dots
post <- data.frame(cbind(alpha.post,beta.post))
ggplot(post, aes(x = alpha.post, y =beta.post)) + geom_density2d() +ylim(-10,40) +xlim(-4,10) +coord_fi
```




```
ggplot(post, aes(x = alpha.post, y =beta.post)) + geom_point()+ylim(-10,40) +xlim(-4,10)+coord_fixed(ra
```



! Caption for the image

Our new priors seem to align our scatter and contour plot with that of the previous uniform priors. Perhaps there is more of a variance towards the top of the ellipse compared to when we use the normal to sample our priors. Applying the context, our prior assumptions do not matter much, as we will continue the predict the same amount of deaths depending on log-dosage and number of animals.

Question 1

####(a)

input data (y-vector)

determine recognize that the posterior can be seperate as so.

$$P(\mu, \sigma^2|data) = P(\mu|\sigma^2, data) \cdot P(\sigma^2|data)$$

Generate σ^2 from $Sc.Inv - \chi^2$(n-1, sˆ2)

use σ^2 to generate μ from N($\bar{y}$, $\frac{\sigma^2}{n}$)

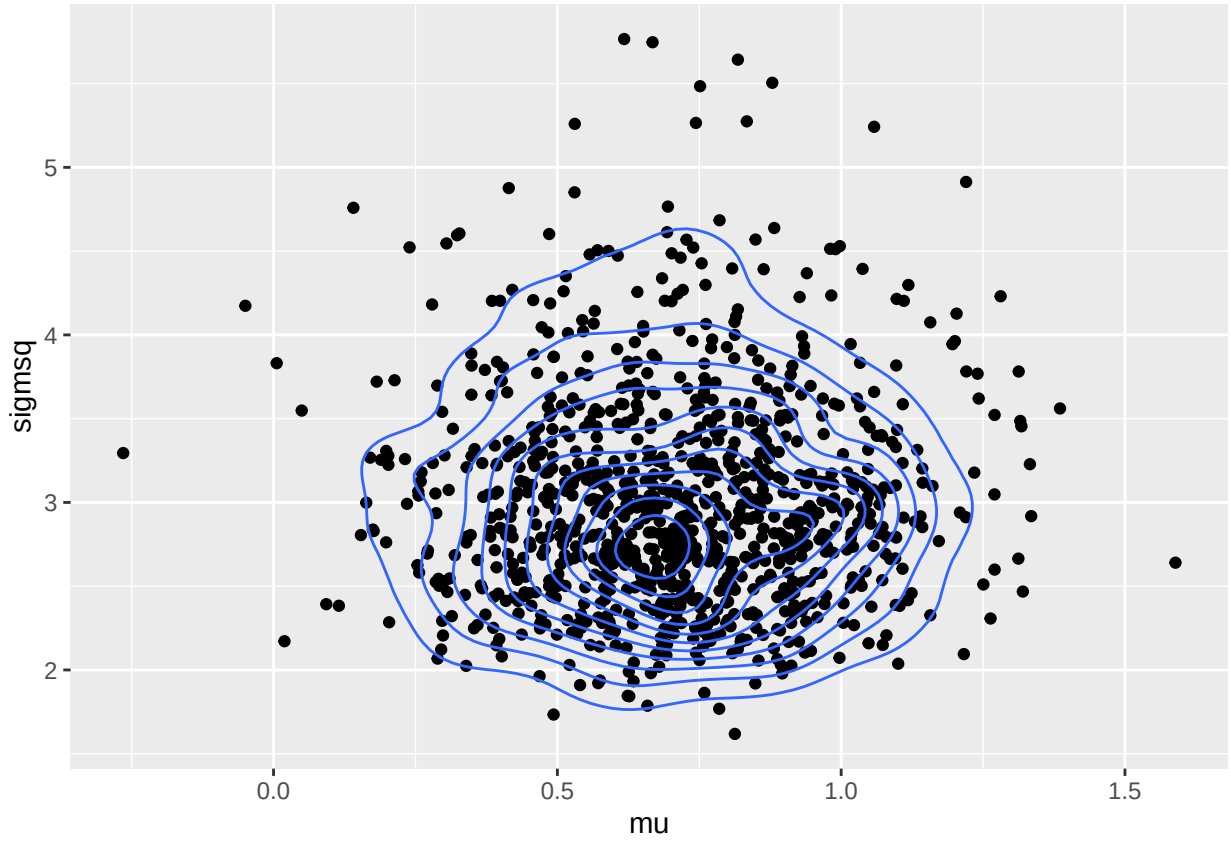
do the last two steps B amount of times

####(b)

```
set.seed(88899)
# grab some data
n = 50
y = rnorm(n,1,2)
ybar = mean(y)
```

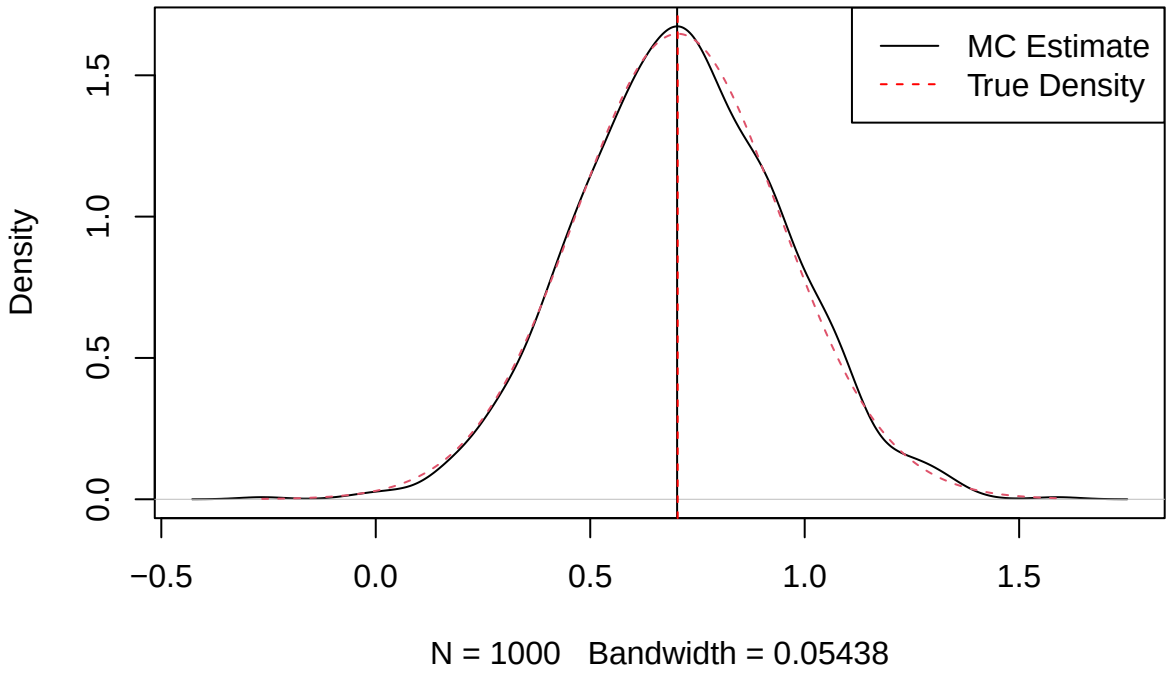
```r
sy = sd(y)
# grab sigma-squareds from scaled inverse chisquared
B = 1000
sigmsq <- (n-1) * sy^2/rchisq(B,n-1)
# grab mus from normal
mu <- rnorm(B,ybar,sqrt(sigmsq/n))
```

```r
post <- data.frame(cbind(mu,sigmsq))
ggplot(post, aes(x = mu, y =sigmsq)) + geom_point() + geom_density2d()
```

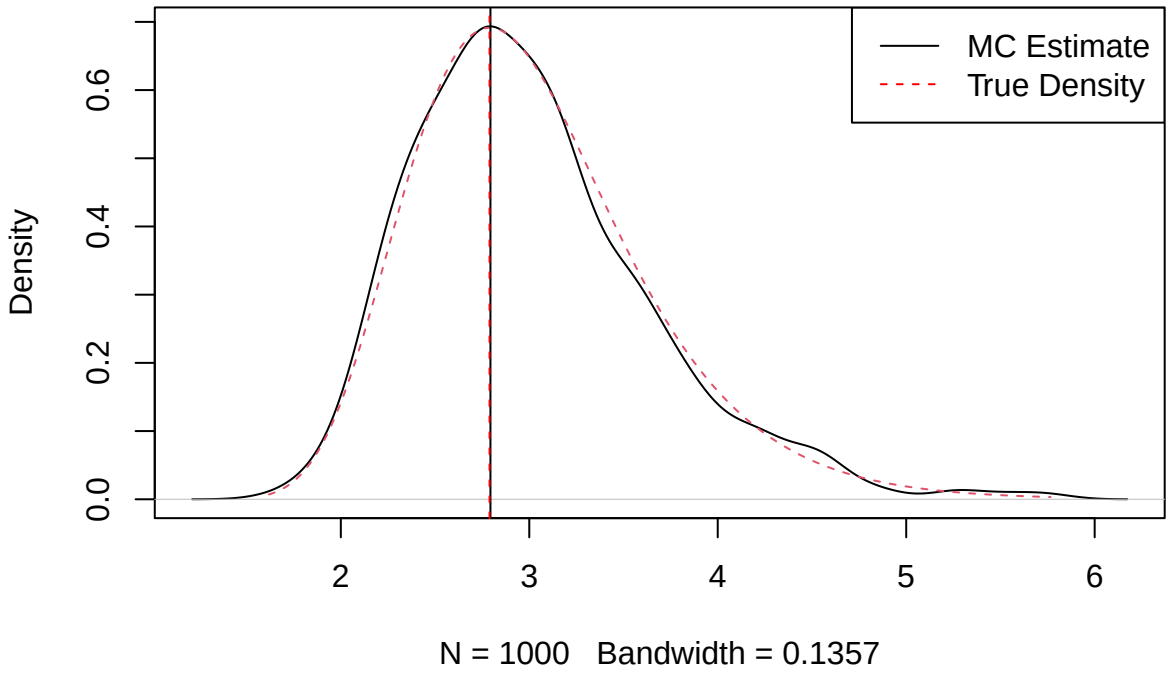


####(c)

```r
# Plot Marginal Posterior of mu
plot(density(mu), main = "p(mu|data)")
mu.seq = seq(min(mu), max(mu), length.out = B)
lines(mu.seq, dt.scaled(mu.seq, n-1, ybar, sqrt(sy^2/n)), col = 2, lty = 2)
legend("topright", legend = c("MC Estimate", "True Density"), col = c("black", "red"), lty = c(1, 2))
# get the MAP value
density(mu)$x[density(mu)$y == max(density(mu)$y)] -> v1
abline(v=v1)
# get actual MLE
av1 = mu.seq[dt.scaled(mu.seq, n-1, ybar, sqrt(sy^2/n)) == max(dt.scaled(mu.seq, n-1, ybar, sqrt(sy^2/n
abline(v = av1, col = "red", lty = 2)
```




```
# Plot Marginal Posterior of sigma
plot(density(sigmsq), main = "p(sigma2|data)")
sigmsq.seq = seq(min(sigmsq), max(sigmsq), length.out = B)
lines(sigmsq.seq, dinvchisq(sigmsq.seq, n-1, sy^2), col = 2, lty = 2)
legend("topright", legend = c("MC Estimate", "True Density"), col = c("black", "red"), lty = c(1, 2))
# get the MAP value
density(sigmsq)$x[density(sigmsq)$y == max(density(sigmsq)$y)] -> v2
abline(v=v2)
# get actual MLE
av2 = sigmsq.seq[dinvchisq(sigmsq.seq, n-1, sy^2) == max(dinvchisq(sigmsq.seq, n-1, sy^2))]
abline(v = av2, col = "red", lty = 2)
```




```
# get absolute differences
abs(av1 - v1); abs(av2 - v2)
```

```
## [1] 0.001222942
```

```
## [1] 0.004788182
```

μ MC estimate flattens out so I went ahead and tried to look for an actual number by essentially looking for the mode. I eventually decided to so this for both estimators and both True and MC estimates. At the end I took the absolute difference and because μ has the smaller difference, I took that to be the one that is better estimate. Nevertheless, they both get relatively close.

####(d)

```
# get y's
ypredpost = c()
for(i in 1:B){
 ypredpost[i] = rnorm(1,mean=mu[i],sd = sqrt(sigmsq[i]))
}
```

recall from hw2 that

$$P(\mu, \sigma^2|y_i) \propto (\sigma^2)^{-2} \prod_{i=1}^{n} \left[(\sigma^2)^{1/2} \exp\left(-\frac{1}{2} \left(\frac{y_i - \mu}{\sigma} \right)^2 \right) \right]$$

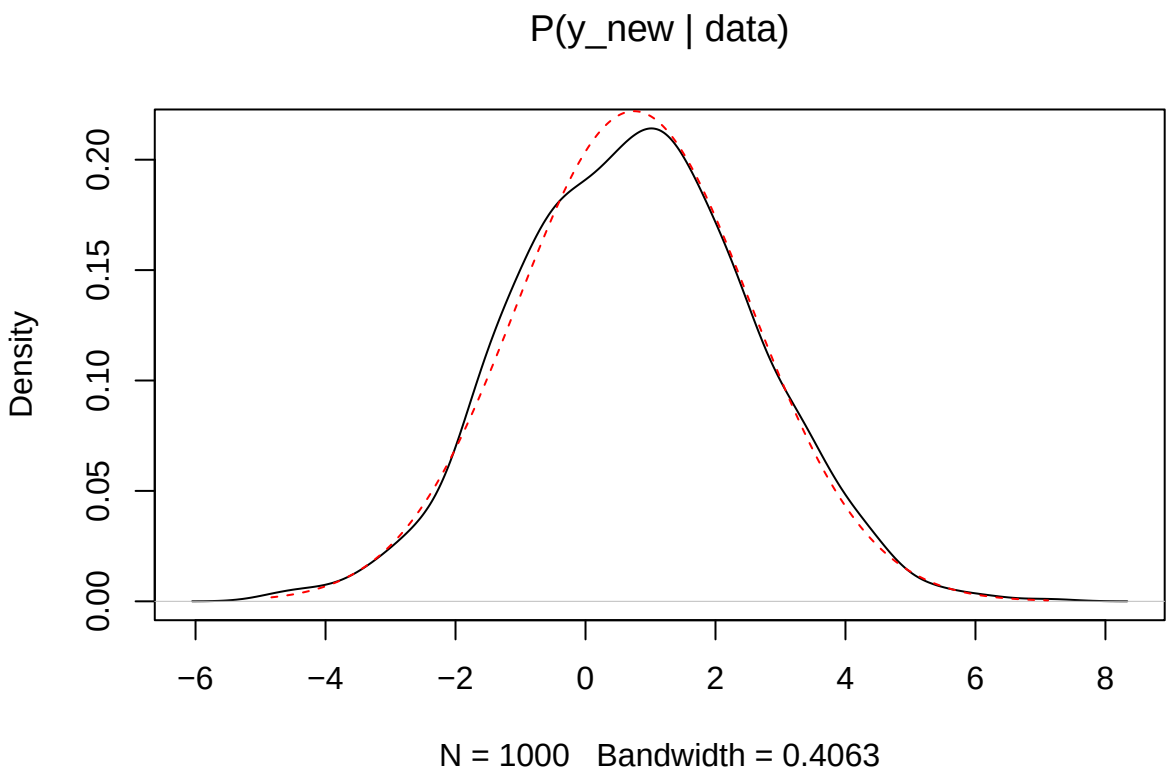
so that

$$P(\tilde{y}|y_i) = \int_{R^+} \int_{R} N(\tilde{y}|\mu, \sigma^2) P(\mu, \sigma^2|y_i) d\mu d\sigma^2$$

$$= t_{n-1}(\bar{y}, s_y^2/n)$$

which is the t-distribution

```
# plot density of MC samples
plot(density(ypredpost), main = expression(paste('P(y_new | data)', sep ="")))
# overlay "true" values
y.seq = seq(min(ypredpost),max(ypredpost),length.out = B)
lines(y.seq,dnorm(y.seq,mean(ypredpost),sd(ypredpost)),col = "red", lty = 2)
```



####(e)

The conditionals

$$\mu|\sigma^2, data \sim N(\mu, \sigma^2/\kappa_n)$$
$$\sigma^2|\mu, data \sim Sc.Inv - \chi^2(\nu_n', \sigma_n^2)$$

This is possible because we did it in class. I am going to write the algorithm in latex

Gibbs implies the following

1] $\sigma_b^2 \sim P(\sigma^2|\mu_{b-1}, data)$ 2] $\mu_b \sim P(\mu|\sigma_b^2, data)$ 3] Move back to 1 and restart

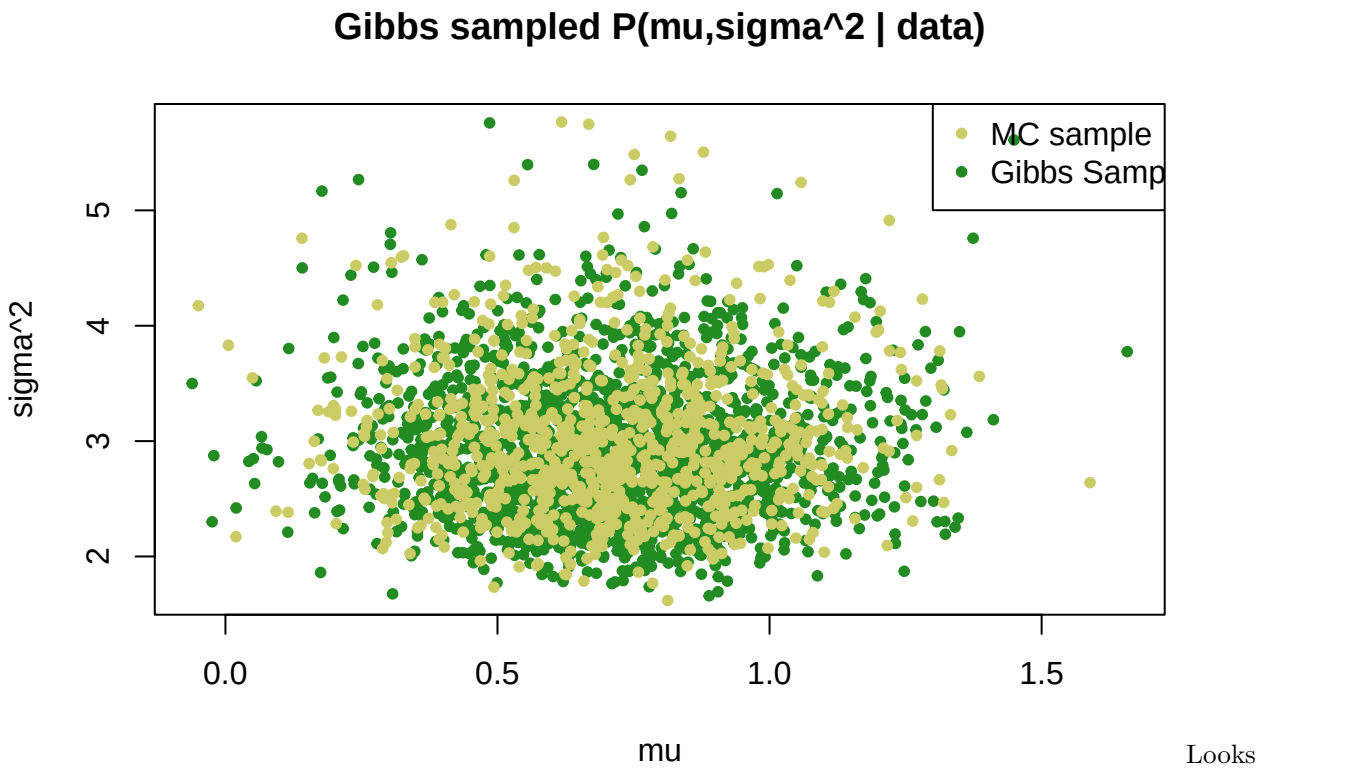
to code this, we need to use the "full posterior conditional distributions" for μ and σ^2, the algorithm becomes

1] get the variance. (a.) $\frac{k_0(\mu_{b-1}-\mu_0)^2+v_0\sigma_0^2+(n-1)*s_y^2+n(\bar{y}-\mu_0)^2}{v_n'} \rightarrow \sigma_n^2\prime$ (b.) $\sigma_b^2|\mu, data \sim Sc.Inv - \chi^2(\nu_n', \sigma_n^2\prime)$
2] get the expected value $\mu_b \sim P(\mu|\sigma_b^2, data)$ $\leftarrow$ feed this value back into (1a).

```
# I will pick initial values.
# It will only be simple.
k0 = 1
mu0 = 1
nu0 = 1
sig20 = 1
```

```r
mu_n <- (k0/(k0 + n) * mu0) + (n/(k0 + n) *
ybar)
k_n <- k0 + n
nu_n.p <- nu0 + n + 1
#
## Posterior samples of mu and sigma2 initialize
mu.post.gibbs <- numeric()
sig2.post.gibbs <- numeric()
mu.post.gibbs[1] <- mean(y)
B <- 2000

### Running the Gibbs Sampler
for (b in 2:(B + 1)) {
sig2n.p <- (k0 * (mu.post.gibbs[(b - 1)] - mu0)^2 + nu0 * sig20 + (n - 1) * sy^2 + (n *(ybar - mu.post.g
sig2.post.gibbs[(b)] <- rinvchisq(1, df = nu_n.p, scale = sig2n.p) #sig2| mu,data
mu.post.gibbs[(b)] <- rnorm(1, mean = mu_n, sd = sqrt(sig2.post.gibbs[(b)]/k_n)) #mu|sig2,data
}
```

```r
plot(mu.post.gibbs,sig2.post.gibbs,
     main = "Gibbs sampled P(mu,sigma^2 | data)",
     xlab = "mu", ylab = "sigma^2",
     col = "forestgreen", pch = 20)
points(mu,sigmsq, pch = 20, col = rgb(0.8,0.8,0.4,1))
legend(1.3,6,c("MC sample","Gibbs Sample"), col = c(rgb(0.8,0.8,0.4,1),"forestgreen"), pch = 20)
```



Looks like the sampling method are almost identical. MC sampling was much easier to do; that is perhaps why we will be utilizing it in the next exercise.

Question 2

```
# artichoke data
y = c(1.8,1.6,1.4,1.6,1.4,1.5,1.2)
```

####(a)

```
# hybrid sampling system

beta = numeric()
alpha = numeric()
alpha[1] = 1
beta[1] <- rgamma(1,alpha[1]*n + 1, 0.8 + sum(y))
n = length(y)
s.tune = 3
accept = 0

#pull an alpha out
#  make functions
Prior.alpha = function(x){
  dgamma(x,2,0.5)
}
Likeli = function(data,A,B){
  prod(dgamma(data,A,B))
}
#J.b = function(data,A){dlnorm(data,A,s.tune)}
J.b = function(x,A){dgamma(x,A*s.tune,s.tune)}
B=200000
############  Gibb + MH Loop ############
for(b in 2:(B+1)){
#pull a beta out
beta[b] <- rgamma(1,alpha[b-1]*n + 1, 0.8 + sum(y))

#  propose an alpha
#rlnorm(1,alpha[b-1],s.tune) -> alpha.star
rgamma(1,alpha[b-1]*s.tune,s.tune) -> alpha.star
# create r
num = Likeli(y,alpha.star,beta[b])*Prior.alpha(alpha.star)/J.b(alpha.star,alpha[b-1])
denom = Likeli(y,alpha[b-1],beta[b])*Prior.alpha(alpha[b-1])/J.b(alpha[b-1],alpha.star)
r = num/denom
if(runif(1) < min(1,r)){
 alpha[b] = alpha.star
 accept = accept + 1
} else {
  alpha[b] = alpha[b-1]
}
}
accept/B
```

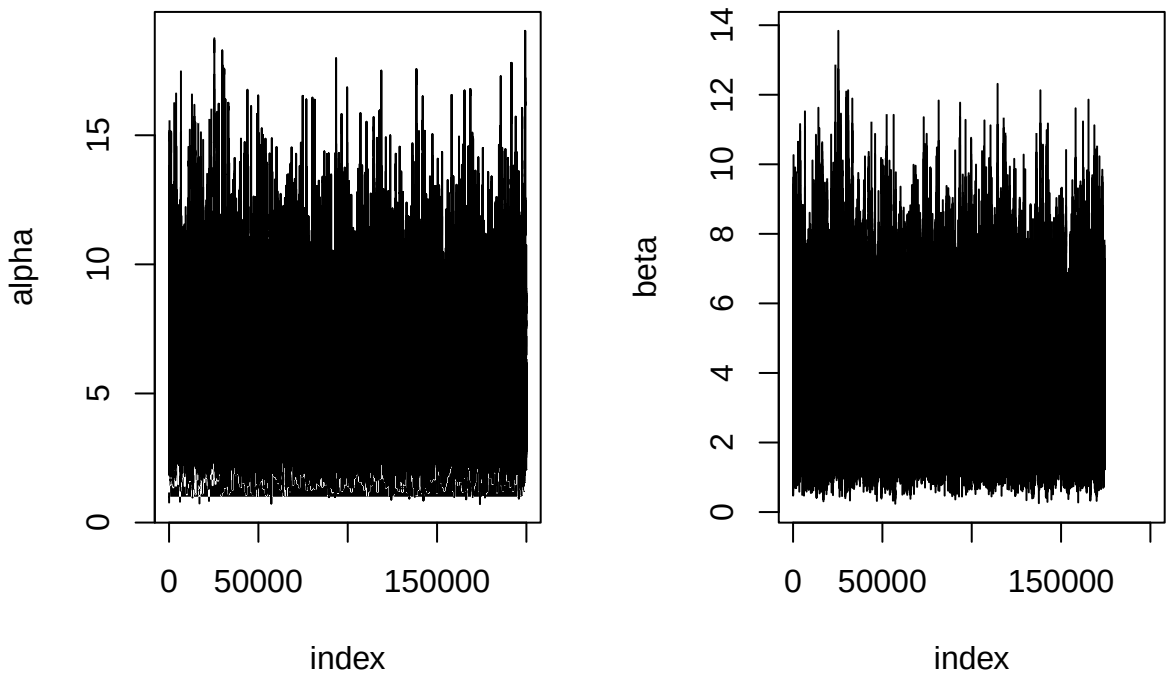
```
## [1] 0.55461
```

```
###################################
```

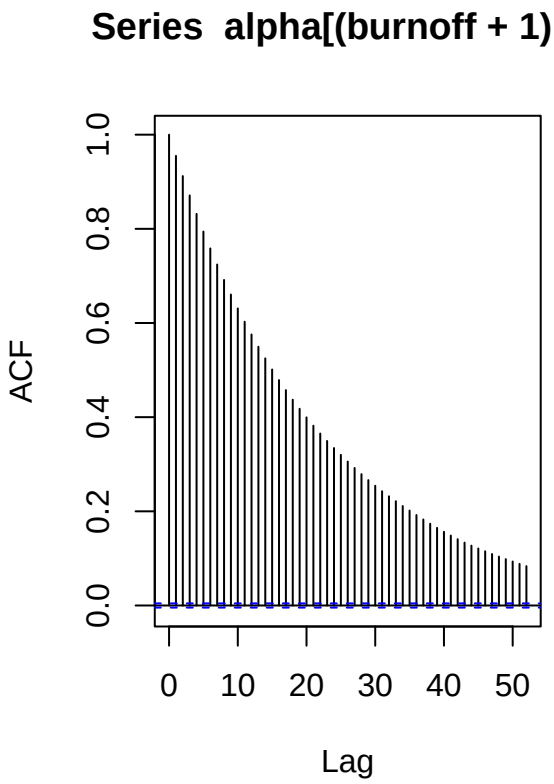
I chose to propose from a Gamma distribution as the LogNorm gave me an acceptance rate of less than 0.5%.

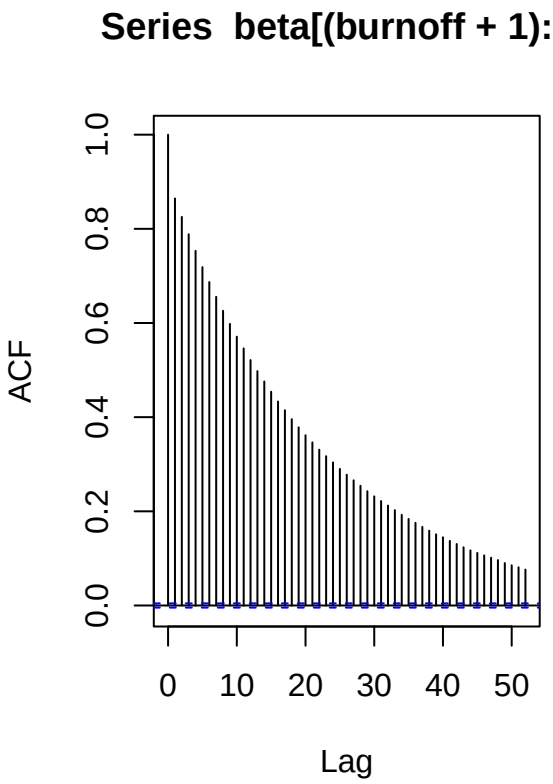
####(b.)

```r
## trace plots
par(mfrow = c(1,2))
plot.ts(alpha,xlab = "index")
plot.ts(beta,xlab = "index")
```

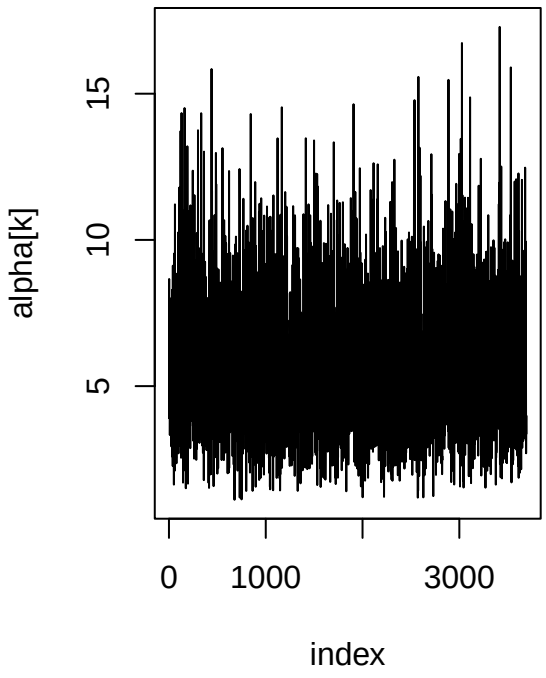


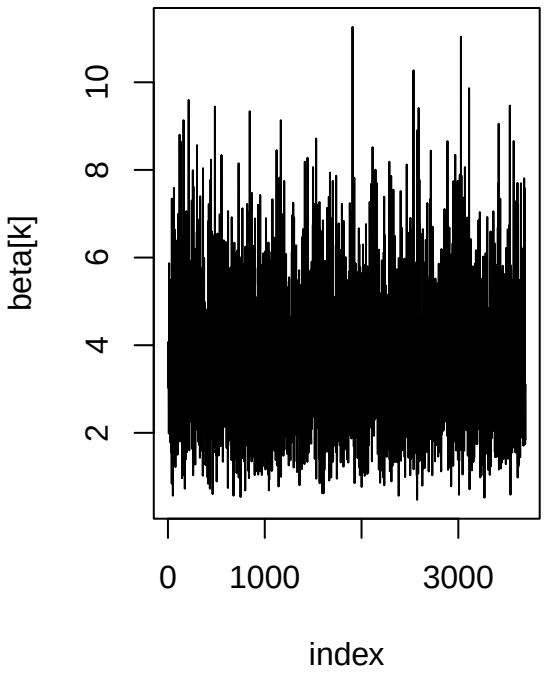

```r
##autocorrelation plots
burnoff = 8000
par(mfrow = c(1,2))
acf(alpha[(burnoff+1):B])
acf(beta[(burnoff+1):B])
```



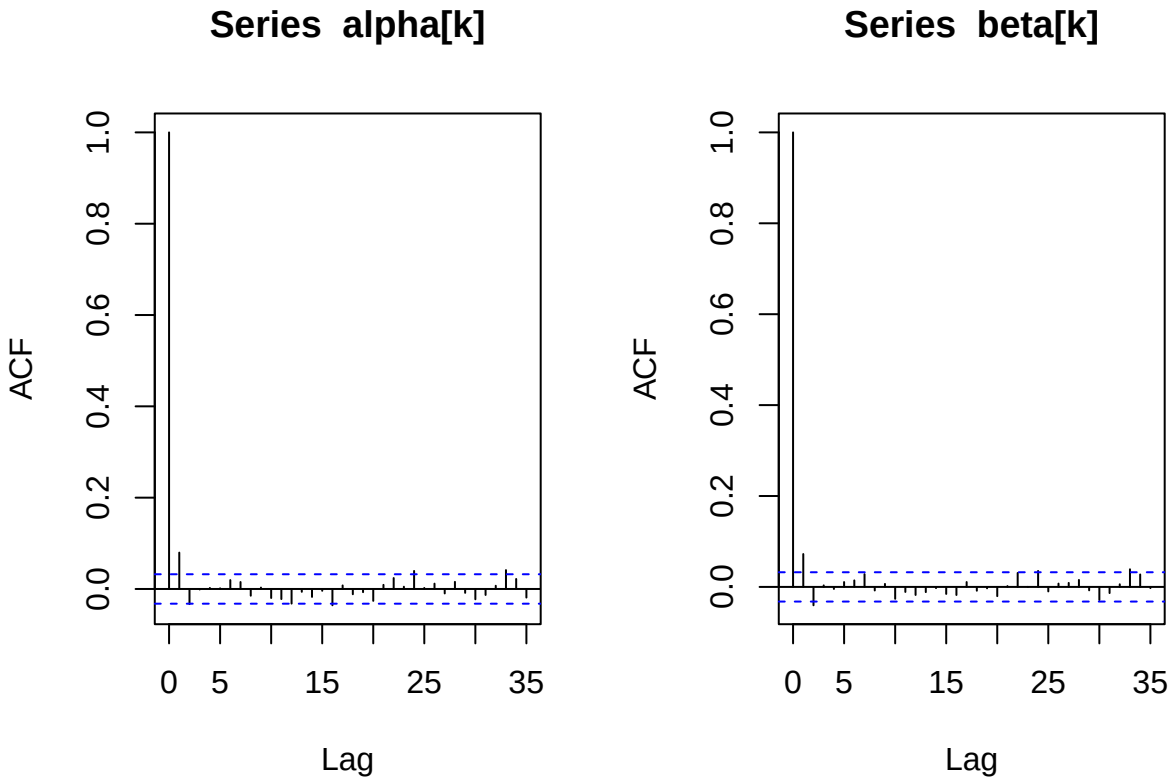



```
# grab every 52nd lag
k = 52*(1:B)
k = k[k < B & k >burnoff]
par(mfrow = c(1,2))
plot.ts(alpha[k],xlab = "index")
plot.ts(beta[k],xlab = "index")
```





```r
acf(alpha[k])
acf(beta[k])
```

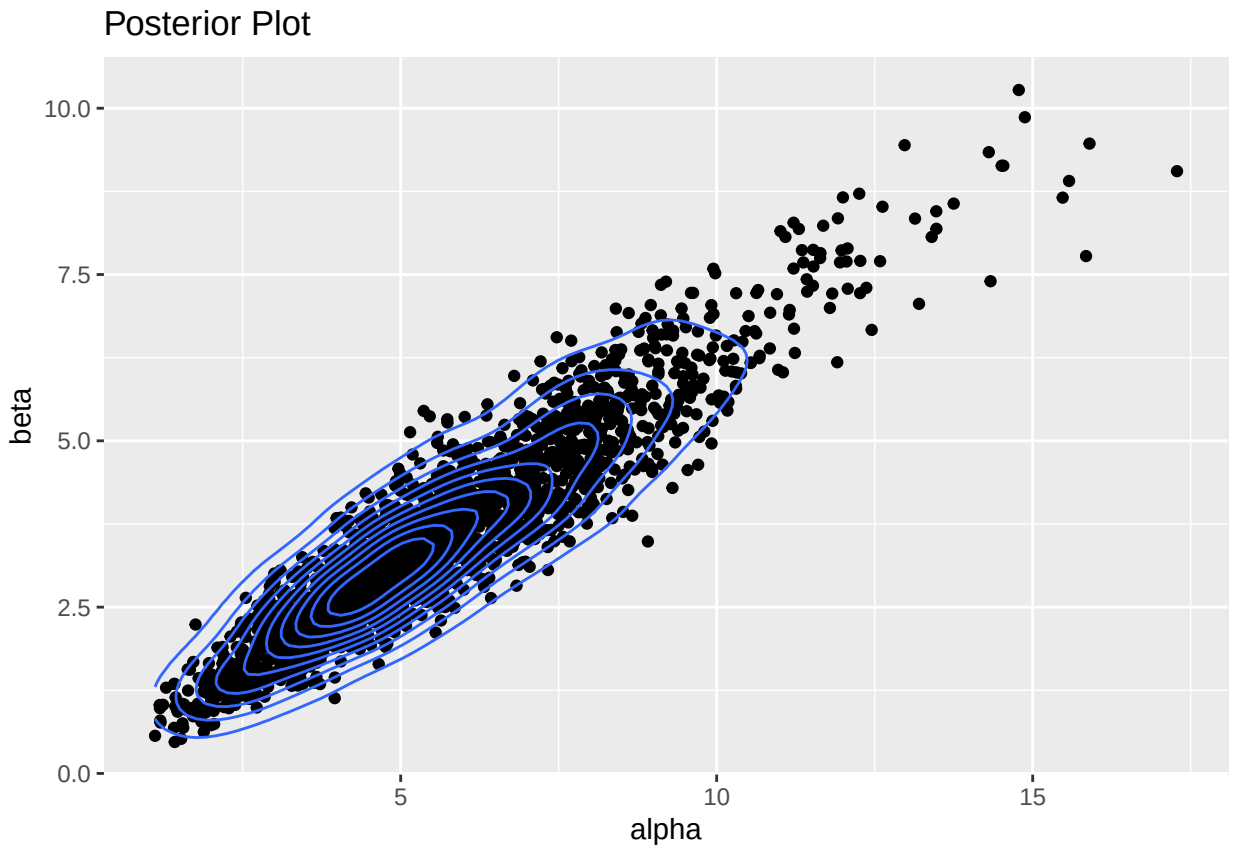



```r
sprintf("I have obtained an acceptance rate of %f and I have burned off %d while also thinning by takin
```

```
## [1] "I have obtained an acceptance rate of 0.554610 and I have burned off 8000 while also thinning by
```

####(d)

```r
# sample 2000
k.i = sample(k, size = 2000)
post <- data.frame(cbind(alpha[k.i],beta[k.i]))
ggplot(post, aes(x = alpha[k.i], y =beta[k.i])) + geom_point() + geom_density2d() + ylab("beta") + xlab
```




```
alpha[k.i] -> alpha
beta[k.i] -> beta
al.MAP = density(alpha)$x[density(alpha)$y == max(density(alpha)$y)]
be.MAP = density(beta)$x[density(beta)$y == max(density(beta)$y)]
al.MAP
```

```
## [1] 4.875447
```

```
be.MAP
```

```
## [1] 3.112147
```

```
quantile(alpha,.025);quantile(alpha,.975)
```

```
##     2.5%
## 1.988586
```

```
##    97.5%
## 11.15699
```

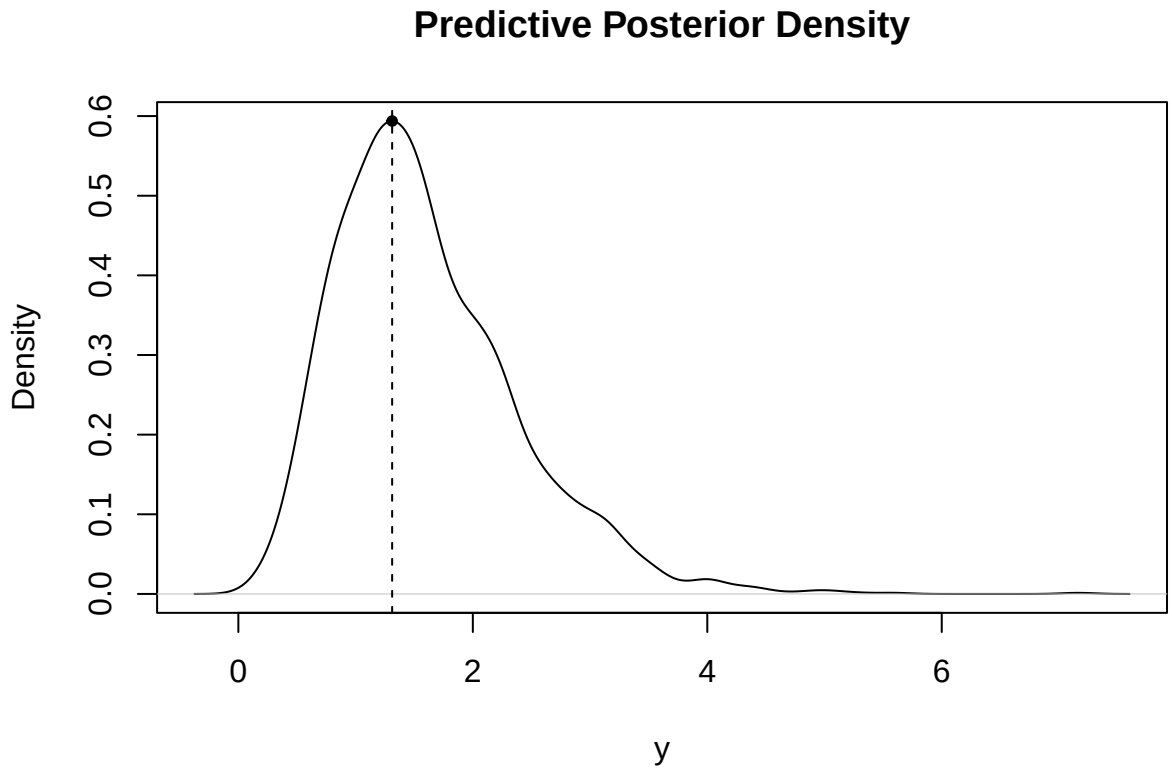
```
quantile(beta,.025);quantile(beta,.975)
```

```
##     2.5%
## 1.169748
```

```
##    97.5%
## 7.223074
```

	MAP	Cred. Int.
α	5.031591	(2.143953,10.78323)
β	3.113442	(1.278739,6.860119)

```
ynews = c()
for(i in 1:2000){
  ynews[i] = rgamma(1,alpha[i],beta[i])
}
ydens = density(ynews)
ynewMAP = ydens$x[ydens$y == max(ydens$y)]
plot(ydens,
     main = "Predictive Posterior Density",
     xlab = "y")
points(ynewMAP,ydens$y[ydens$y == max(ydens$y)],pch = 20)
abline(v = ynewMAP,lty = 2)
```



(e)

```
sprintf("I have found the MAP of Ynew to be %f with a 95-percent credible interval of (%f,%f)",ynewMAP,
```

```
## [1] "I have found the MAP of Ynew to be 1.311727 with a 95-percent credible interval of (0.480522,3.4
```