MCMC Sampling

Lennox Garay

2023-10-16

```
library(boot)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

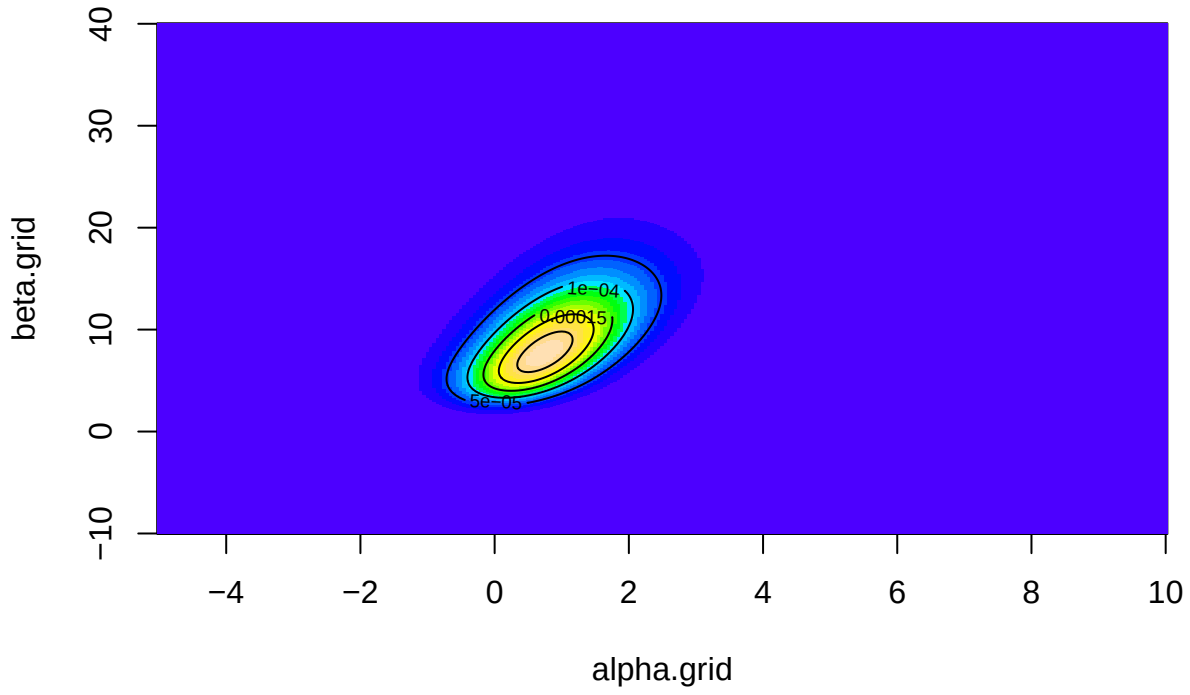
```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

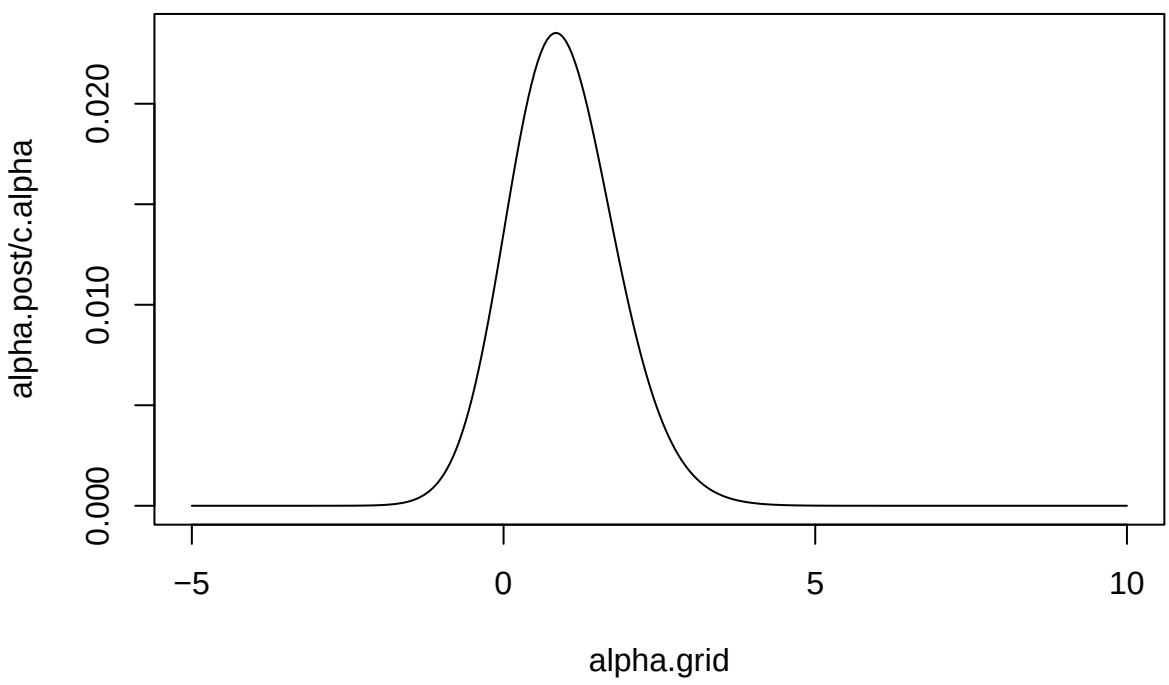
```
library(mvtnorm)
```

```
set.seed(538)
mu <- matrix(c(0,10), ncol=1)
sig <- matrix(c(4,10,10,100) ,nrow=2 ,byrow = TRUE)

post <- function(alpha,beta){
  x = c(-0.86, -0.3, -0.05, 0.73)
  n <- c(5, 5, 5, 5)
  y <- c(0, 1, 3, 5)
  likelihood = invlogit(alpha + beta * x)
  p = prod(likelihood^y * (1 - likelihood)^(n - y))*dmvnorm(c(alpha, beta), as.vector(mu), sig)
  return(p)
}


post = function(alpha, beta) {
x = c(-0.86, -0.3, -0.05, 0.73) #the data
n = c(5, 5, 5, 5)
y = c(0, 1, 3, 5)
alpha.prior = dnorm(alpha, mean = 0, sd = 2)
```

```
beta.prior = dnorm(beta, mean = 10, sd = 10)
temp = inv.logit(alpha + beta * x)
p = prod(temp^y * (1 - temp)^(n - y))*alpha.prior*beta.prior
return(p)
}

alpha.grid <- seq(-5, 10, 0.05)
a.n <- length(alpha.grid)
beta.grid <- seq(-10, 40, 0.1)
b.n <- length(beta.grid)
post.grid <- matrix(0, nrow = a.n, ncol = b.n)

alpha.post = seq(0, length.out=a.n)
for (i in 1:a.n) {
  for (j in 1:b.n) {
post.grid[i, j] <- post(alpha.grid[i], beta.grid[j])
}

  alpha.post[i] <- sum(post.grid[i, ])
}
c.joint <- sum(post.grid) #Finding the normalizing constants
c.alpha <- sum(alpha.post)

image(alpha.grid, beta.grid, post.grid/c.joint, col = topo.colors(20))
contour(alpha.grid, beta.grid, post.grid/c.joint,
add = TRUE)
```

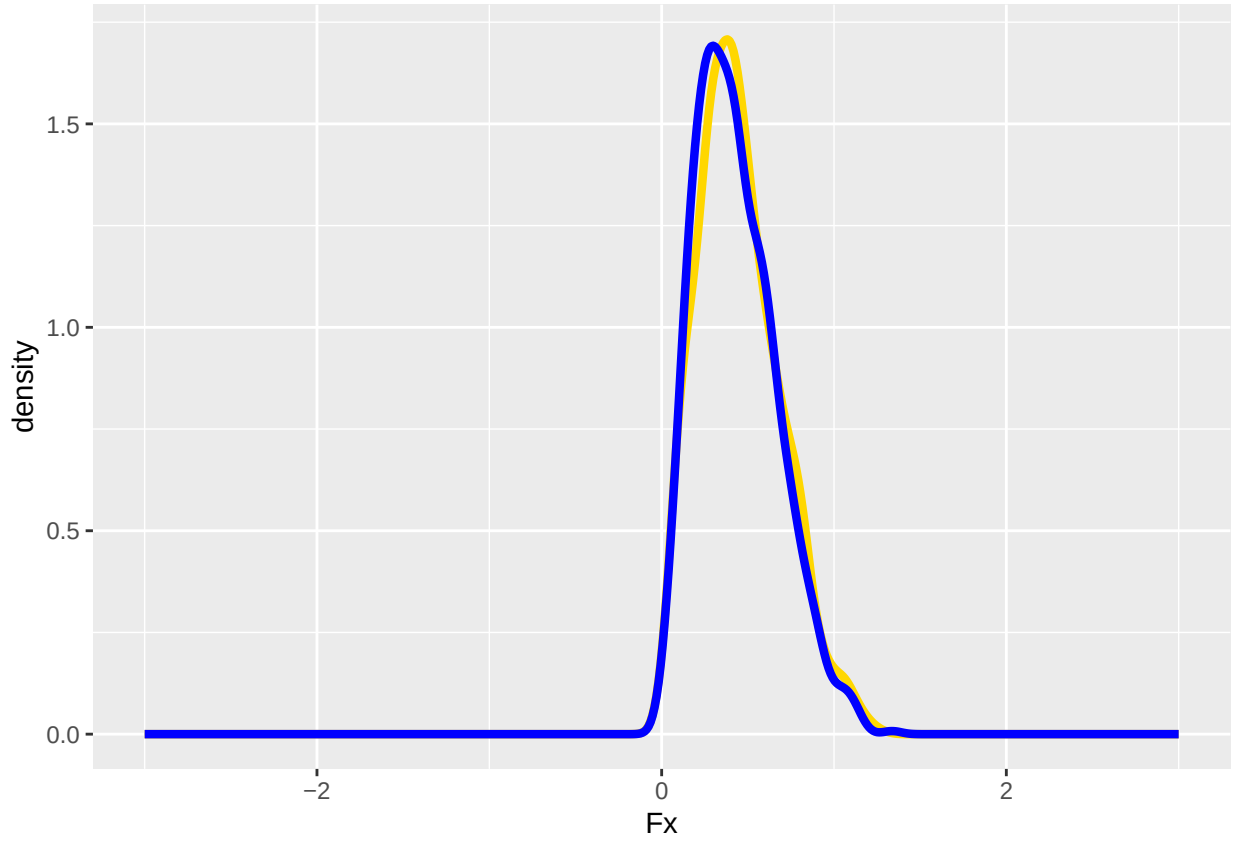


```
plot(alpha.grid, alpha.post/c.alpha, type = "l")
```



1 (a): Inverse CDF Sampling

```
n = 1000
U = runif(n, 0,1)

iweib = function(U,alpha,beta){
  ex = (1/alpha)
  temp = log(1-U)
  q = ((-1*(temp))^ex)* beta
  return(q)
  }

Fx = iweib(U,alpha=2, beta=0.5)
truth = rweibull(n,2,0.5)
grid = seq(-50,50, length.out=1000)
df = data.frame(U,Fx, truth,grid)

ggplot(df, aes(x=Fx)) + geom_density(aes(x=Fx), colour='gold', lwd=1.5) + geom_density(aes(x=truth), col
```

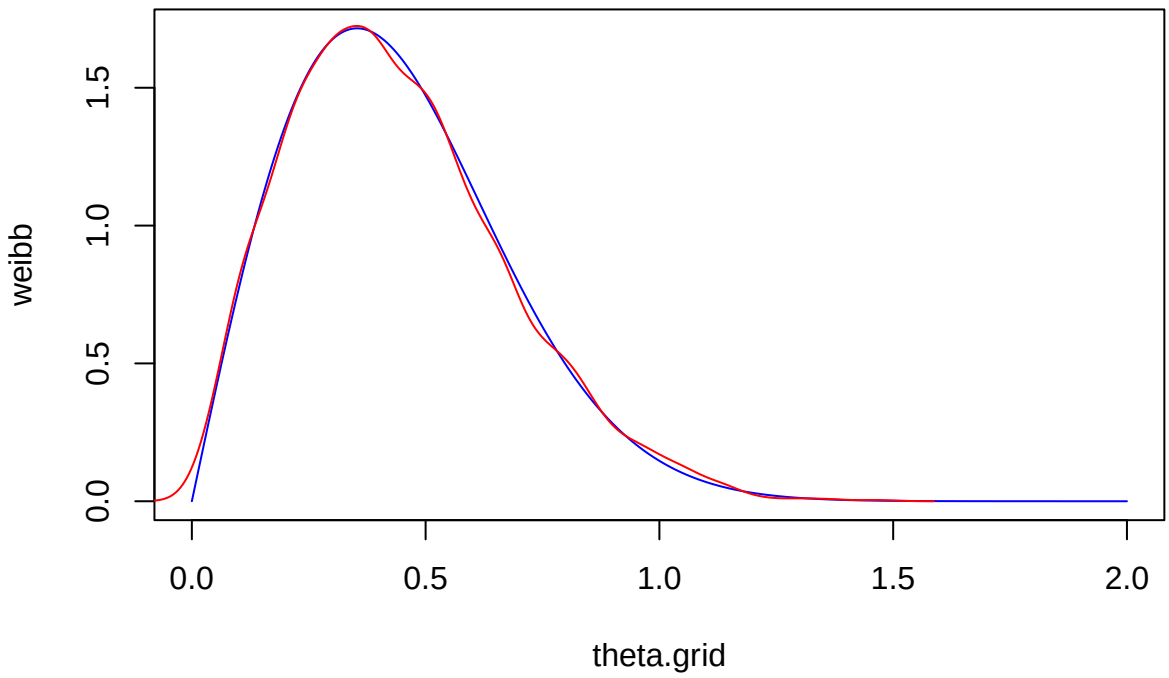


1(b):

```
alpha = 2
beta = 0.5
k = alpha
N = 10000
U = runif(N,0,1)
Y = rexp(N, rate=1/beta)
fy = dweibull(Y,alpha,scale=beta) ## "f" in f/g
g = k*dexp(Y,rate=1/beta) #this is our envelope
accepted = Y[U <= (fy/g)]

length(accepted)/N ## %accepted
```

```
## [1] 0.5037
```

```
theta.grid = seq(0,2, length.out=1000)

weibb = dweibull(theta.grid,alpha,scale=beta)

plot(theta.grid,weibb, type='l', col='blue')
lines(density(accepted), type='l', col='red')
```



2 (a):

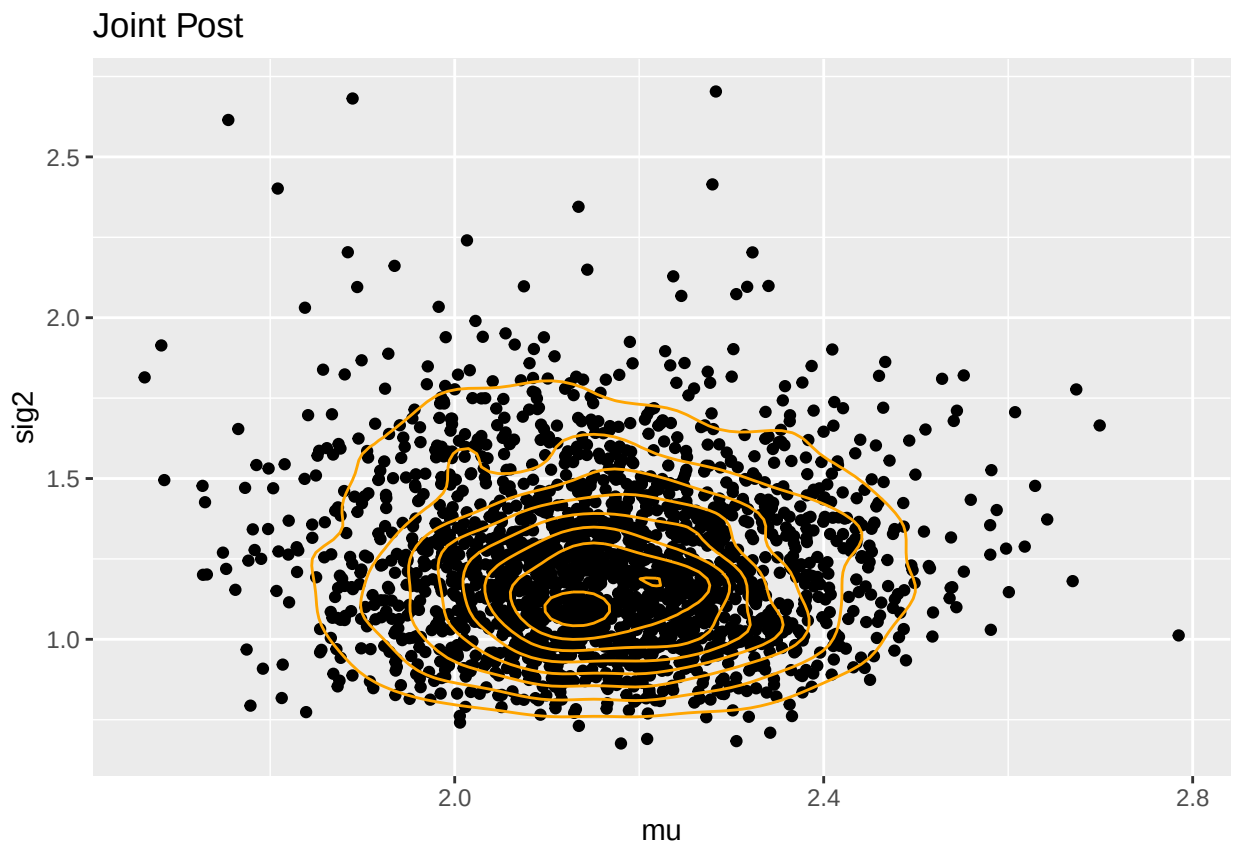
Algorithm:

1) Simulate some data from normal (this generates y_i.
2) Calculate sample variance and sample mean from n data.
3) We draw N random samples from $\sigma^2|y \sim Sc.Inv.\chi^2(n-1, s^2)$

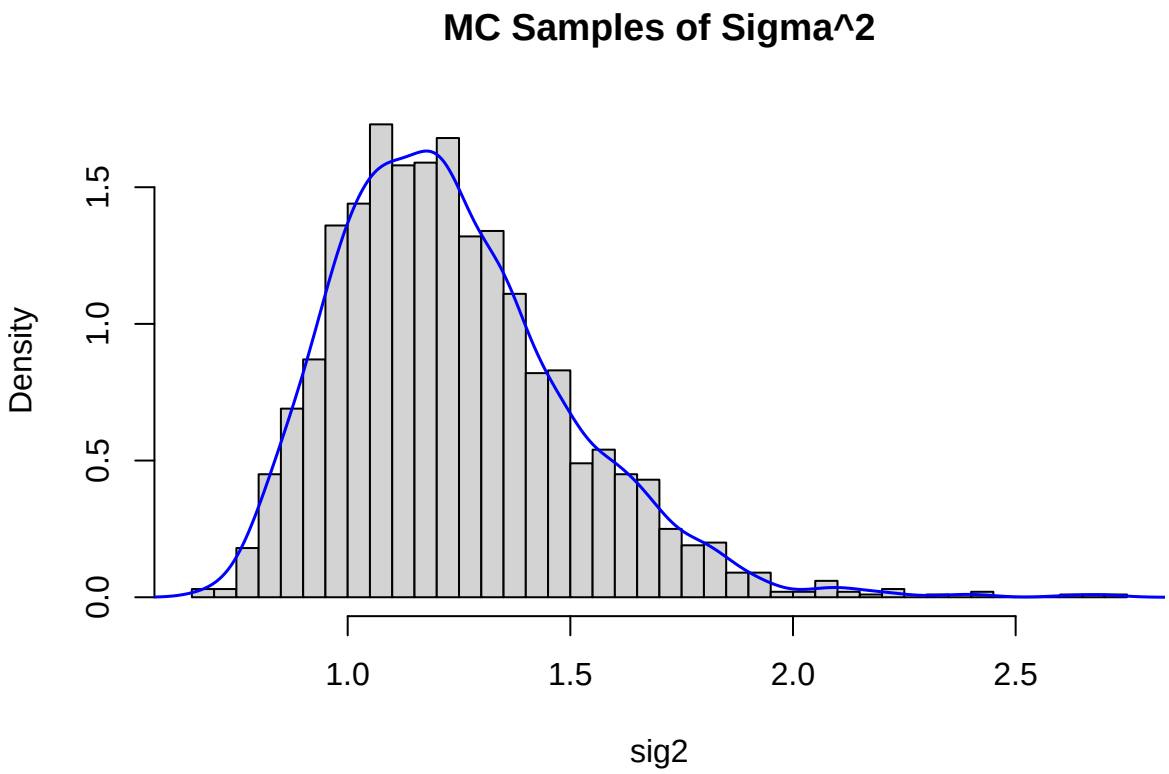
4) We know that $\mu|\sigma^2, y \sim N(\bar{y}, \frac{\sigma^2}{n})$. (of course this relies on the sampled values of σ^2 in step 3.)

MC Integration

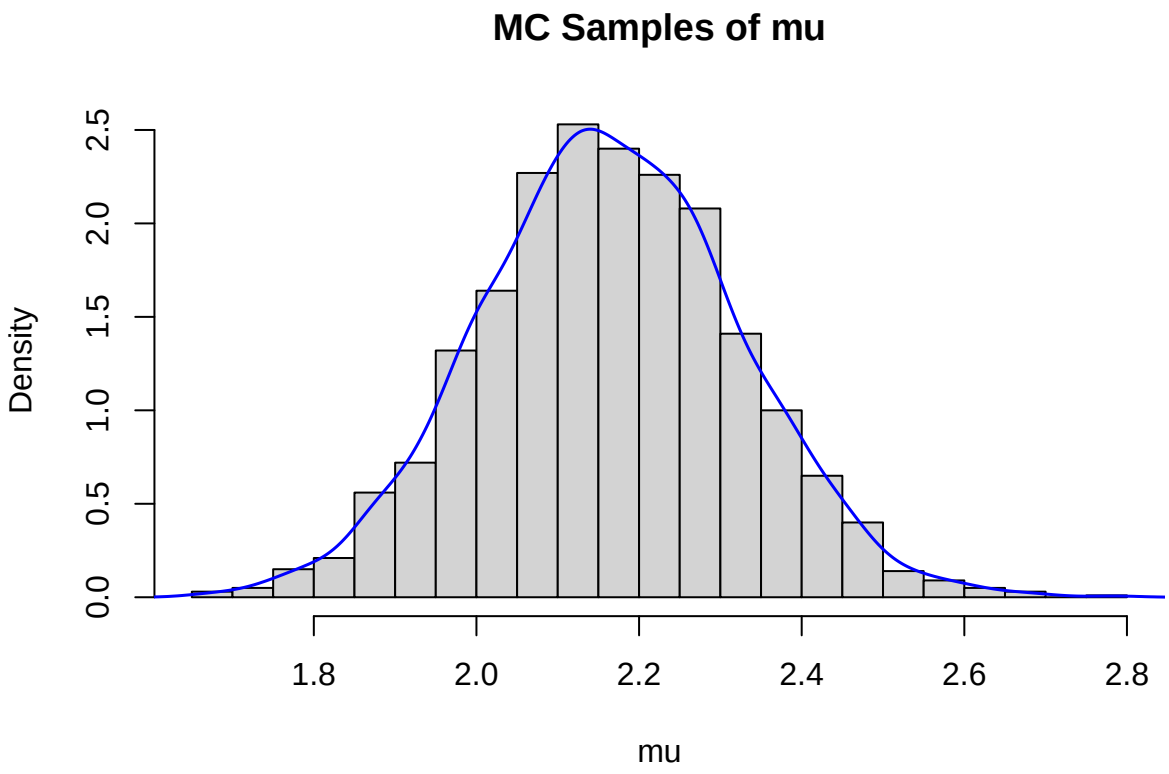
```
n = 50
y = rnorm(n,2,1)

s = sd(y)
s2 = s^2
ybar = mean(y)

N = 2000
sig2 = rinvchisq(N, df=n-1, scale=s2)
mu = rnorm(N, mean=ybar, sd = sqrt(sig2/n))

post = data.frame(mu,sig2)

ggplot(data=post, aes(x=mu, y=sig2)) + geom_point() + geom_density_2d(colour='orange') + ggtitle('Joint
```




```
df2 = data.frame(sig2)

griddy = seq(0,3, length.out=length(sig2))

hist(sig2, breaks = 30, prob=T,main = 'MC Samples of Sigma^2')
lines(density(sig2), type='l', col = 'blue', lwd = 1.5)
```




```
hist(mu, breaks=30, prob=T, main = 'MC Samples of mu')
lines(density(mu), type='l', col='blue',lwd=1.5)
```




```
Mode <- function(x) { ## Taken off the web; tried an R package but it gave me the wrong answer.
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

cbind(mu=Mode(mu),sig=Mode(sig2))
```

```
##            mu      sig
## [1,] 2.405056 1.204039
```

We see that the MAP is pretty close to the true parameter values! How fun.

2 (d):

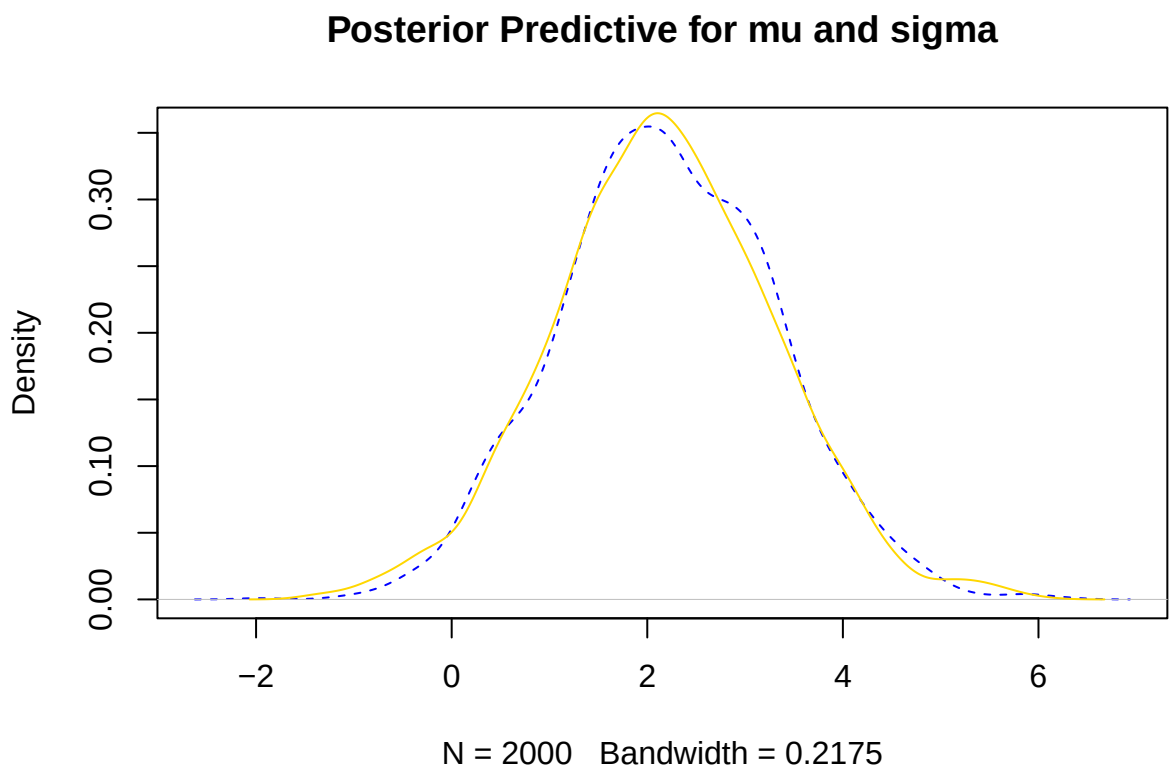
We have that the posterior predictive is:

$\int \int p(\tilde{y}|\mu, \sigma^2) p(\mu, \sigma^2|y) d\mu d\sigma^2$

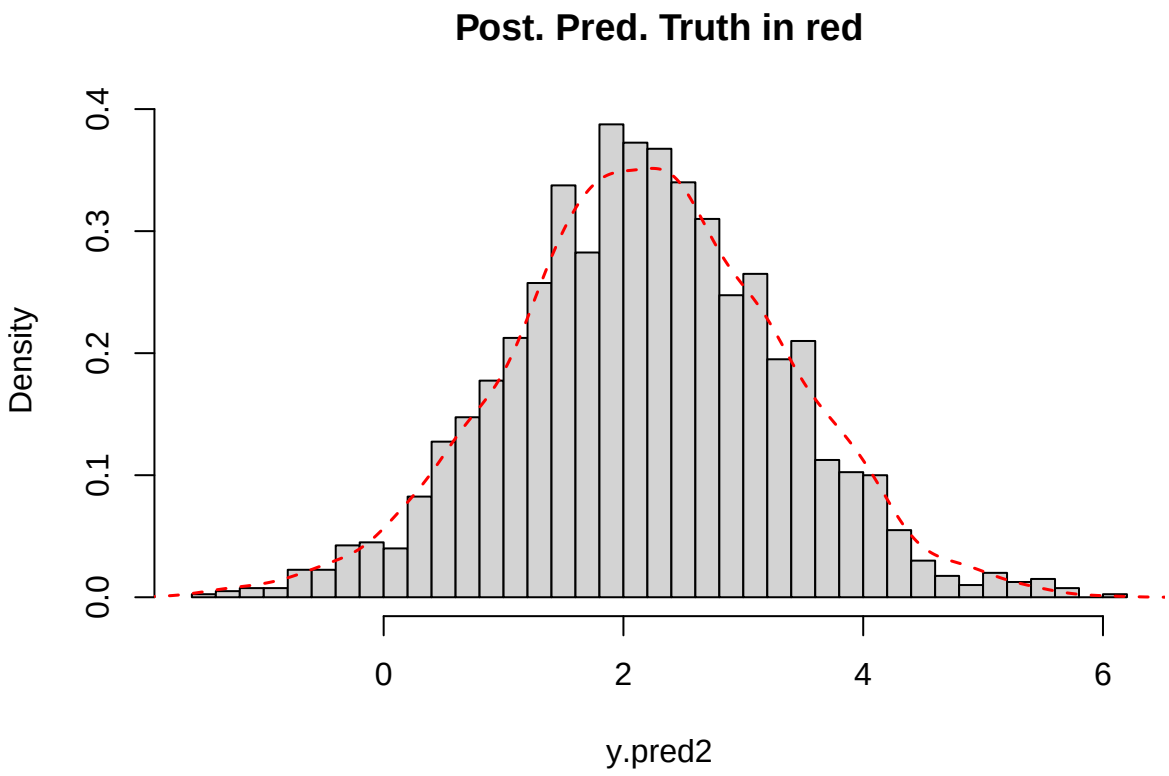
Which we can sample from a Normal distribution using the posterior samples of mu and σ^2 so, y~ $N(\mu_{post}, \sigma^2_{post})$

```
## using these for the posterior predictive
sig.new = sig2
mu.new = mu

y.pred = as.numeric()
for(i in 1:length(sig.new)){
  y.pred[i] = rnorm(1, mean=mu.new[i], sd=sqrt(sig.new[i]))
}
```

```r
## is it possible to do this without a for loop? eh lets try

samps = length(sig2)

y.pred2 = rnorm(n=samps, mean=mu.new, sd=sqrt(sig.new))

plot(density(y.pred), col = 'blue', main = 'Posterior Predictive for mu and sigma', lty='dashed')
lines(density(y.pred2), col = 'gold', main = 'Posterior Predictive for mu and sigma?')
```

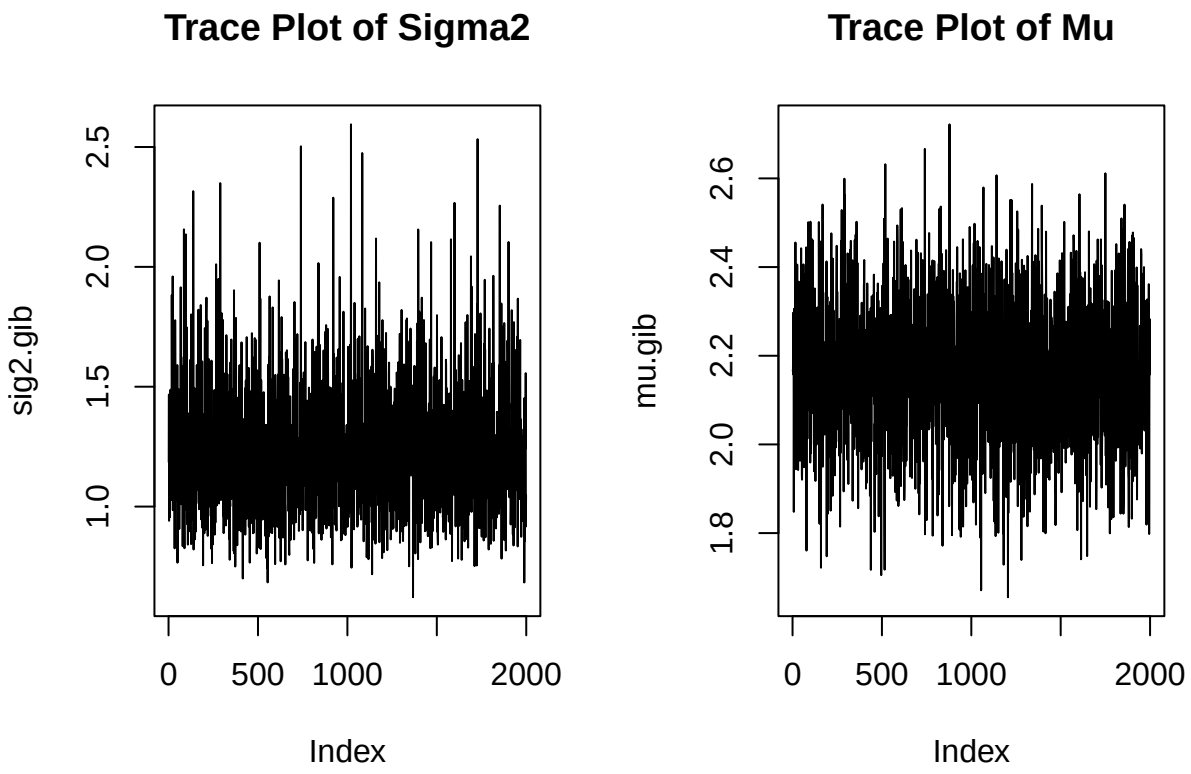



```r
## BEAUTIFUL! NO FOR LOOP NEEDED. We love R vectorization.

## Theoretical Post Pred

sig.real = var(y)

y_samps2 = rt.scaled(2000, df = n-1,mean = ybar, sd = (sqrt(1 + 1/n))*sqrt(sig.real))

hist(y.pred2, breaks=30, main='Post. Pred. Truth in red', prob=T)
lines(density(y_samps2), col = 'red', lty='dashed', lwd=1.5)
```

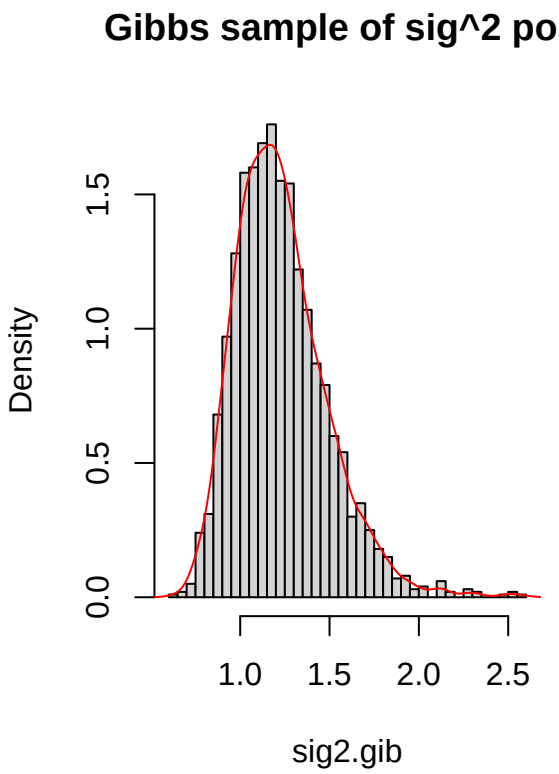


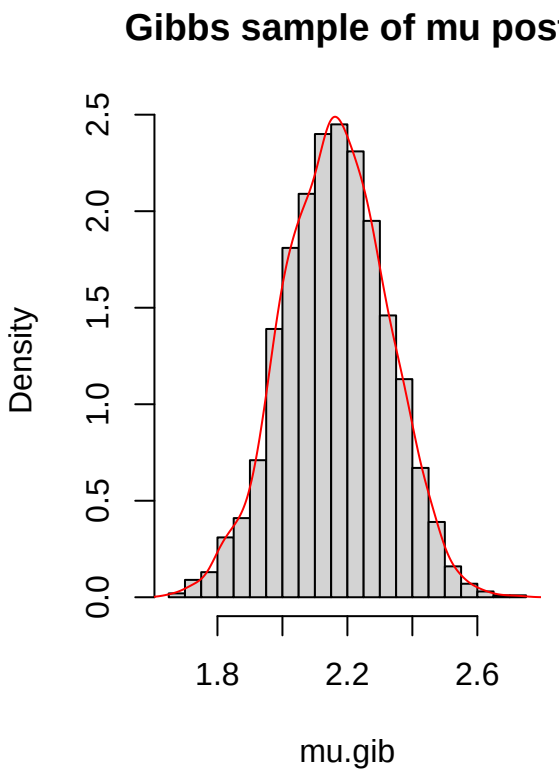
E: Gibbs(?!)

```
B = 2000

mu.gib = rep(0, length(y))
sig2.gib = rep(0, length(y))

mu.gib[1] = ybar ## initialization

for(b in 2:B){
  scaled = sum((y - mu.gib[b-1])^2)/n
  sig2.gib[b-1] = rinvchisq(1,df=n,scale=scaled)
  mu.gib[b] = rnorm(1,ybar, sd=sqrt(sig2.gib[b-1]/n))

}


par(mfrow=c(1,2))
plot(sig2.gib, type="l", main = "Trace Plot of Sigma2")
plot(mu.gib, type = "l", main = "Trace Plot of Mu")
```




```
hist(sig2.gib, breaks=30, main='Gibbs sample of sigˆ2 post', prob=T)
lines(density(sig2.gib), col='red')
hist(mu.gib, breaks=30, main='Gibbs sample of mu post', prob=T)
lines(density(mu.gib), col='red')
```





We were in fact able to use Gibbs sampling on this data. Both MC and Gibbs sampling were pretty good at estimating the true posterior density. It was pretty straight forward, but I think that if MC samples works, then it is best to use that. MC sampling is even simpler than Gibbs.