

## Lab 5: MCMC

Team USA: Yuting Deng, Kendra Gilbertson, Lily Durkee, Bennett Hardy

2022-09-14

Contact: Hermione.Deng@colostate.edu, kendra01@colostate.edu, L.Durkee@colostate.edu, Bennett.Hardy@colostate.edu

### Problem

You will write code using conjugate relationships, also known as Gibbs updates, to draw samples from marginal posterior distributions of a mean and variance.

1. Set the seed for random numbers = 10 in R with `set.seed(10)`.
2. Load the `actuar` library, which contains functions for inverse gamma distributions.
3. Simulate 100 data points from a normal distribution with mean  $\theta = 100$  and variance  $\zeta^2 = 25$ . Call the data set `y`. Be careful here. R requires the standard deviation, not the variance, as a parameter. You will use these “fake” data to verify the Gibbs sampler you will write below. Simulating data is always a good way to test methods. Your method should be able to recover the generating parameters given a sufficiently large number of simulated observations.

I have saved you some time by writing a function called `draw_mean` that makes draws from the marginal posterior distributions for  $\theta$  using a normal-normal conjugate relationship where the variance is assumed to be known. It is vital that you study the `MCMCmath.pdf` notes relative to this function.

4. I have saved you some time by writing a function called `draw_mean` that makes draws from the marginal posterior distributions for  $\theta$  using a normal-normal conjugate relationship where the variance is assumed to be known. It is vital that you study the `MCMCmath.pdf` notes relative to this function.
5. I have also provided a function called `draw_var` that makes draws from the marginal posterior distribution for  $\zeta^2$  using a inverse gamma-normal conjugate relationship where the mean is assumed to be known. Study this function relative to the `MCMCmath.pdf` handout.
6. Check the functions by simulating a large number of data points from a normal distribution using a mean and variance of your own choosing. Store the data points in a vector called `y_check`. Assume flat priors for the mean and the variance. A vague prior for the inverse gamma has parameters  $\alpha_0 = .001$  and  $\beta_0 = .001$ .

```
## $z
## [1] 0.0791588
##
## $mu_1
## [1] 0.001564032
##
## $sigma.sq_1
## [1] 0.002499938
```

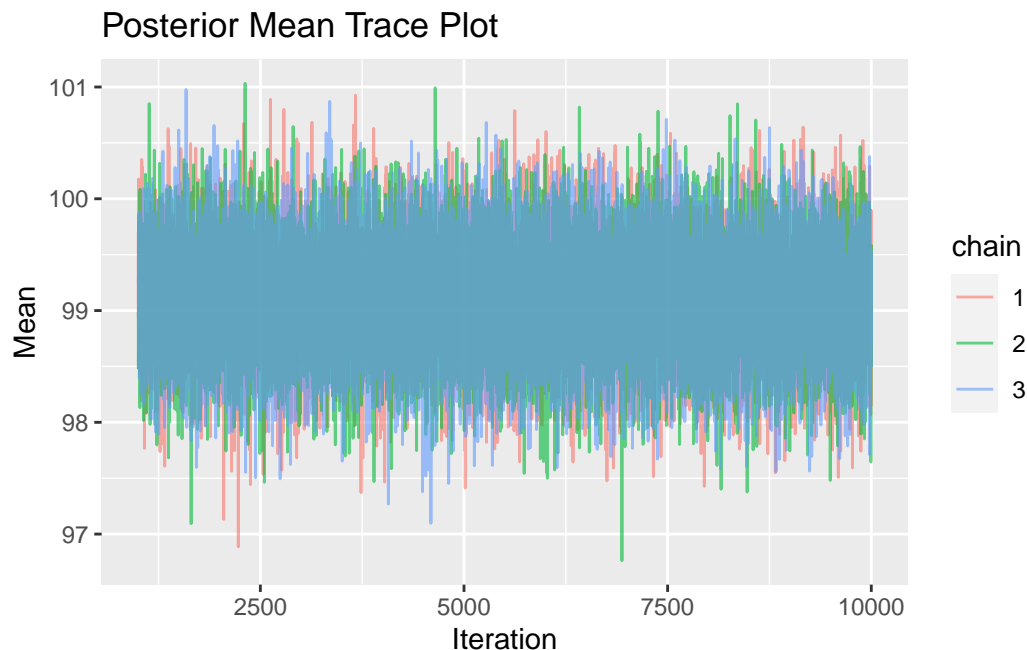
```
## $z
## [1] 9907.478
##
## $alpha_1
## [1] 5000.001
##
## $beta_1
## [1] 50003495
```

## Write a sampler

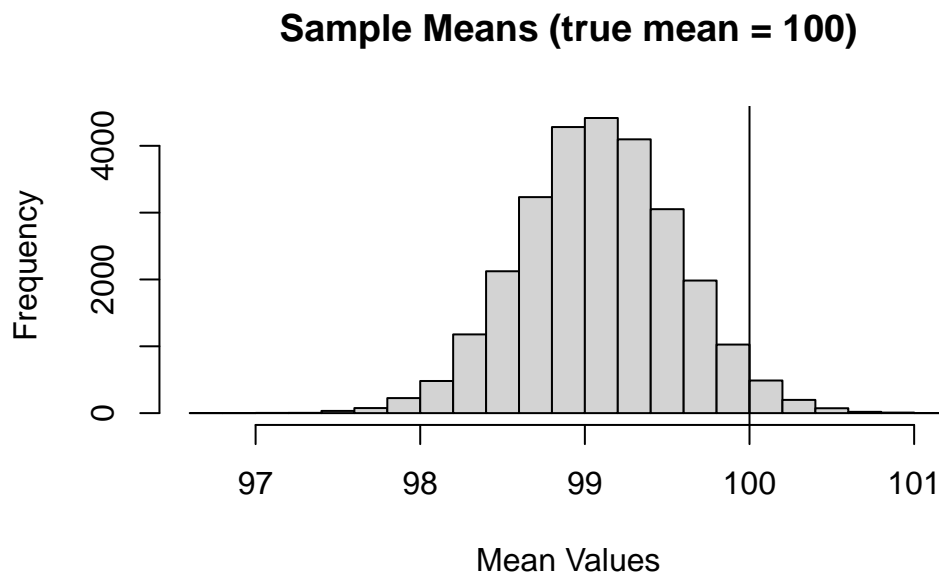
1. Set up a matrix for storing samples from the posterior distribution of the mean. The number of rows should equal the number of chains (3) and number of columns should equal the number of iterations (10,000). Do the same thing for storing samples from the posterior distribution of the variance.
2. Assign initial values to the first column of each matrix, a different value for each of the chains. These can be virtually any value within the support of the random variable, but it would be fine for this exercise to use values not terribly far away from those you used to simulate the data, reflecting some prior knowledge. You might try varying these later to show that you will get the same results.
3. Set up nested for loops to iterate from one to the total number of iterations for each of the three chains for each parameter. Use the conjugate functions `draw_mean` and `draw_var` to draw a sample from the distribution of the mean using the value of the variance at the current iteration. Then make a draw from the variance using the current value of the mean. Repeat. Assume vague priors for the mean and variance:

## Trace plots and plots of marginal posteriors

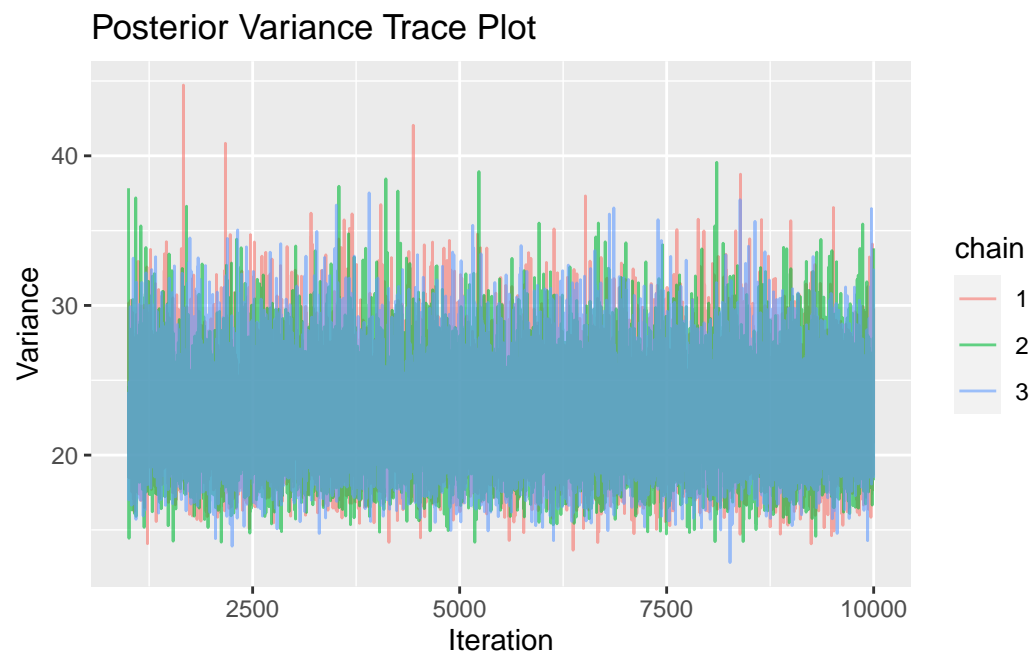
1. Discard the first 1000 iterations as burn-in. Plot the value of the mean as a function of iteration number for each chain. This is called a trace plot.

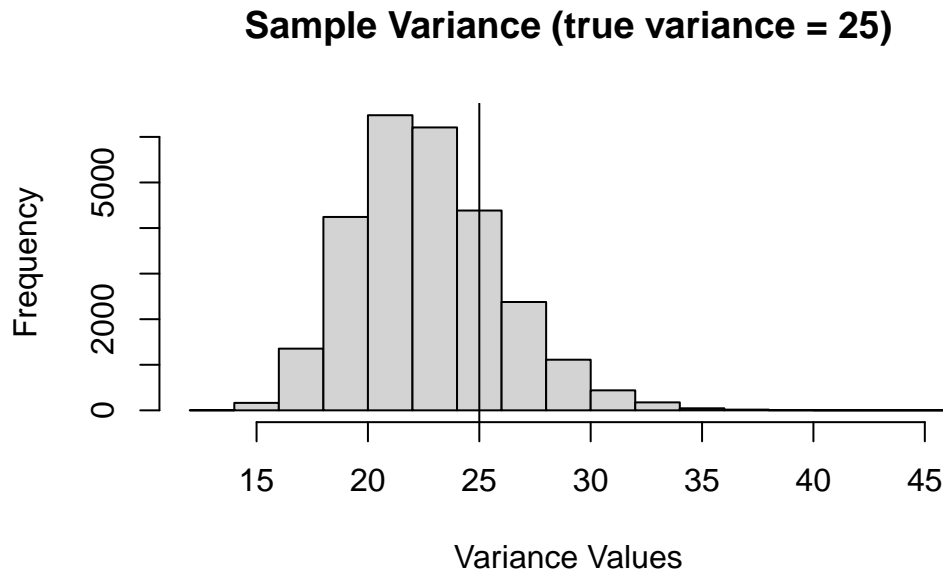


2. Make a histogram of the samples of the mean retained after burn-in including all chains. Put a vertical line on the plot showing the generating value.



3. Repeat steps 1-2 for the variance.





4. For both  $\theta$  and  $\zeta^2$ , calculate the mean of all the chains combined and its standard deviation. Interpret these quantities.

Mean:

```
## [1] 99.08613
```

```
## [1] 0.4788827
```

The average mean of all our 9000 iterations of the mean is 99.1, and its standard deviation is 0.48. The estimate is very precise, and close to our true known mean of 100.

Variance

```
## [1] 22.68649
```

```
## [1] 3.304163
```

The average mean of all our 900 iterations of the variance is 22.69, and its standard deviation is 3.3. The estimate is very precise, and close to our true known variance of 25.

5. Compare the standard deviation of the posterior distribution of  $\theta$  with an approximation using the standard deviation of the data divided by the square root of the sample size. What is this approximation called in the frequentist world?

```
## [1] 0.4788827
```

```
## [1] 0.4706179
```

The approximation in the frequentist world is called “standard error”.

6. Vary the number of values in the simulated data set, e.g.,  $n = 10, 100, 1,000, 10,000$ . We do not exactly recover the generating values of  $\theta$  and  $\varsigma^2$  when  $n$  is small. Why? The mean of the marginal posterior distribution of the variance is further away from its generating value than the mean is. Why? Try different values for set seed with  $n = 100$  and interpret the effect of changing the random number sequence.

A.  $n=10$

Average mean and standard deviation of mean

```
## [1] 97.35436
```

```
## [1] 2.403836
```

Average variance and standard deviation of variance

```
## [1] 45.82257
```

```
## [1] 29.29341
```

B.  $n=100$

Average mean and standard deviation of mean

```
## [1] 100.6783
```

```
## [1] 0.5256242
```

Average variance and standard deviation of variance

```
## [1] 27.09358
```

```
## [1] 4.351669
```

C.  $n=1000$

Average mean and standard deviation of mean

```
## [1] 99.85155
```

```
## [1] 0.1584277
```

Average variance and standard deviation of variance

```
## [1] 25.47553
```

```
## [1] 1.111865
```

C.  $n=10000$

Average mean and standard deviation of mean

```
## [1] 100.0269
```

```
## [1] 0.05050042
```

Average variance and standard deviation of variance

```
## [1] 25.28128
```

```
## [1] 0.3518398
```

E. Use a different value for set seed with n=100

Average mean and standard deviation of mean

```
## [1] 99.59813
```

```
## [1] 0.4632297
```

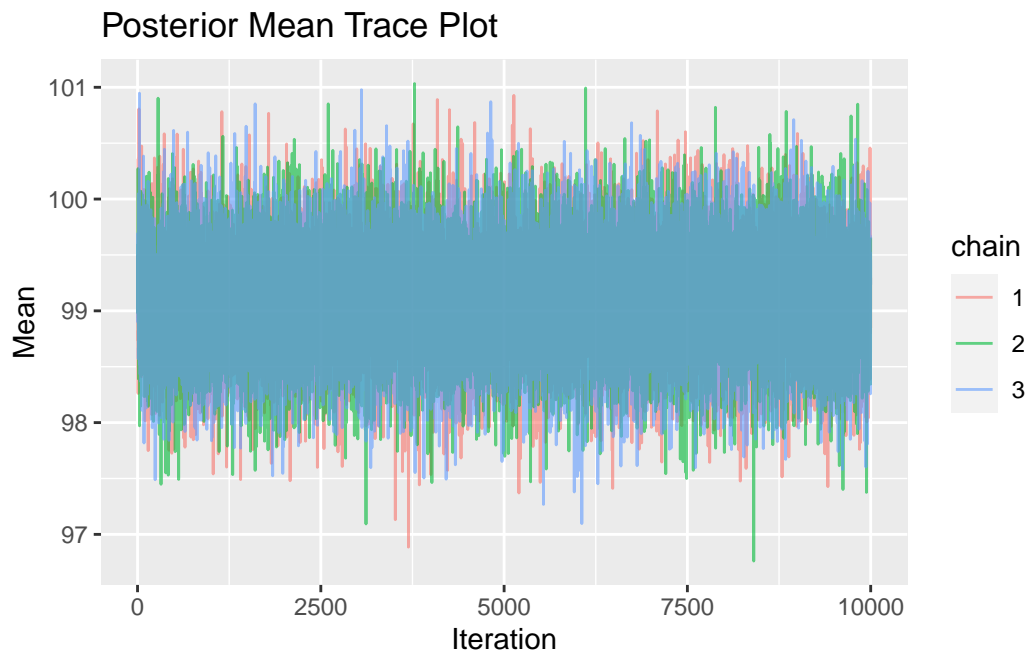
Average variance and standard deviation of variance

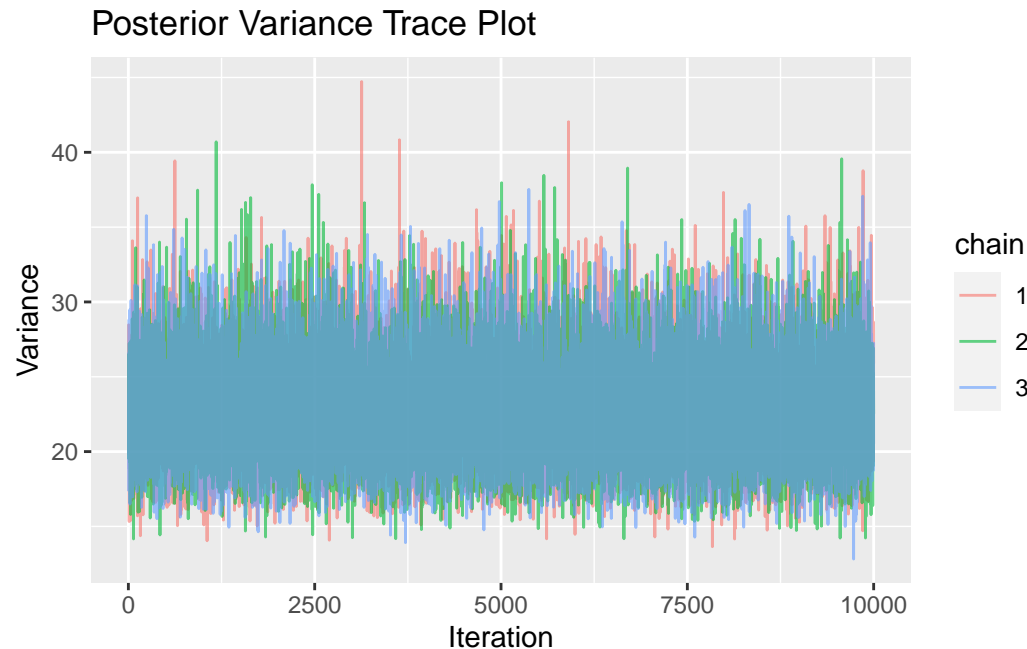
```
## [1] 22.45903
```

```
## [1] 3.423258
```

When setting a different seed value and sample size is relatively small ( $n=100$ ), we saw a different mean and variance of the posterior distribution from the ones with `seed.value = 10`, both different from the generating values. It is because each sample is achieving a different result in stochastic process, thus the mean and variance of the posterior distribution would be different each time. When sample size is small, it's harder to get to the true value. When sample size is big enough ( $n=10000$ ), the posterior distribution has the mean and variance closer to the generating value. The mean of the marginal posterior distribution of the variance is further away from its generating value than the mean because variance is more sensitive to the dispersed data.

7. Make the `burnin=1` instead of 1000. Does this change your results? Why or why not?





```
## [1] 99.08613
```

```
## [1] 0.4788827
```

```
## [1] 22.68649
```

```
## [1] 3.304163
```

Despite reducing the burnin to one, both our posterior mean and variance quickly converged. This means that Gibbs update is very effective. Their averages were very close to the true values, and the standard deviation for both estimates was small. This is because we are still using a large sample size of 10000; it is a large enough sample to make up for a short burnin period.

8. Reverse the order of the conjugate functions in step 3 of the Writing a Sampler section so that the variance is drawn first followed by the mean. Does this reordering have an effect on the posteriors? Why or why not?

```
## [1] 99.08613
```

```
## [1] 0.4788827
```

```
## [1] 22.68649
```

```
## [1] 3.304163
```

Changing the order within the sampler does not effect the posteriors because samples are drawn from the full-conditionals. The only significant change is that we're using the posterior mean initial value instead of the variance initial value, and from the second iteration on they're each samplin from each other. This is too small to make a difference, but we're also dropping the first iteration during burning, so it isn't included in our results.

## Code

```
1 knitr::opts_chunk$set(  
2   echo = FALSE,  
3   fig.height = 3.5,  
4   fig.width = 5.5,  
5   message = FALSE,  
6   warning = FALSE,  
7   attr.source = ".numberLines"  
8 )  
9 library(ggplot2)  
10 library(gridExtra)  
11 library(dplyr)  
12 library(ggpubr)  
13 library(mathjaxr)  
14 library(actuar)  
15 set.seed(10)  
16 library(actuar)  
17 y<-rnorm(100,mean=100,sd=5)  
18 # normal likelihood with normal prior conjugate for mean, assuming variance is known  
19 # mu_0 is prior mean  
20 # sigma.sq_0 is prior variance of mean  
21 # varsigma.sq is known variance of data  
22  
23 draw_mean = function(mu_0, sigma.sq_0, varsigma.sq, y){  
24   mu_1=((mu_0 / sigma.sq_0 + sum(y)/varsigma.sq)) / (1/sigma.sq_0 + length(y) / varsigma.sq)  
25   sigma.sq_1 = 1/(1 / sigma.sq_0 + length(y) / varsigma.sq)  
26   z = rnorm(1, mu_1, sqrt(sigma.sq_1))  
27   param = list(z = z, mu_1 = mu_1, sigma.sq_1 = sigma.sq_1)  
28   return(param)  
29 }  
30 # normal likelihood with gamma prior conjugate relationship for variance, assuming mean is known  
31 # alpha_0 is parameter of prior for variance  
32 # beta_0 is parameter of prior for variance  
33 # Note that this uses scale parameterization for inverse gamma  
34  
35 draw_var = function(alpha_0, beta_0, theta, y){  
36   alpha_1 = alpha_0 + length(y) / 2  
37   beta_1 = beta_0 + sum((y - theta)^2) / 2  
38   z = rinvgamma(1, alpha_1, scale = beta_1)  
39   param = list(z = z, alpha_1 = alpha_1, beta_1 = beta_1)  
40   return(param)  
41 }  
42 y_check<-rnorm(10000,0,1)  
43  
44 draw_mean(mu_0=0, sigma.sq_0=100, varsigma.sq=25, y=y_check)  
45  
46 draw_var(alpha_0=.001, beta_0=.001, theta=100, y=y_check)  
47 post_mean<-matrix(NA,nrow=3,ncol=10000)  
48 post_var<-matrix(NA,nrow=3,ncol=10000)  
49 post_mean[,1]<-c(3,4,5)  
50 post_var[,1]<-c(3,4,5)  
51 a=.001
```



```

52 b=.001
53 mu<-0
54 sigma<-100
55
56 for(i in 1:9999){
57   for(x in 1:3){
58     post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i], y=y)$z
59     post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i+1], y=y)$z
60   }
61 }
62 post_mean2<-post_mean[,c(1001:10000)]
63 post_var2<-post_var[,c(1001:10000)]
64
65 df_mean = data.frame(iteration=rep(1:10000, times=3),
66                       chain=as.character(rep(1:3, each=10000)),
67                       mean=c(post_mean[1,], post_mean[2,], post_mean[3,]))
68
69 # mean traceplot
70 ggplot(df_mean %>% filter(iteration>1000))+
71   geom_line(aes(x=iteration, y=mean, group=chain, color=chain), alpha=0.6)+
72   labs(title="Posterior Mean Trace Plot", x="Iteration", y="Mean")
73 # mean histogram
74 allmean<-c(post_mean2)
75 hist(allmean,main="Sample Means (true mean = 100)",
76       xlab="Mean Values")
77 abline(v=100)
78 df_var = data.frame(iteration=rep(1:10000, times=3),
79                     chain=as.character(rep(1:3, each=10000)),
80                     var=c(post_var[1,], post_var[2,], post_var[3,]))
81
82 # variance traceplot
83 ggplot(df_var %>% filter(iteration>1000))+
84   geom_line(aes(x=iteration, y=var, group=chain, color=chain), alpha=0.6)+
85   labs(title="Posterior Variance Trace Plot", x="Iteration", y="Variance")
86 # variance histogram
87 allvar<-c(post_var2)
88 hist(allvar,main="Sample Variance (true variance = 25)",
89       xlab="Variance Values")
90 abline(v=25)
91 mean(allmean)
92 sd(allmean)
93 mean(allvar)
94 sd(allvar)
95 sd(allmean)
96 sd(y)/sqrt(length(y))
97 y<-rnorm(10,mean=100,sd=5)
98
99 post_mean<-matrix(NA,nrow=3,ncol=10)
100 post_var<-matrix(NA,nrow=3,ncol=10)
101
102 post_mean[,1]<-c(3,4,5)
103 post_var[,1]<-c(3,4,5)
104

```

```

105 a=.001
106 b=.001
107 mu<-0
108 sigma<-100
109
110 for(i in 1:9){
111     for(x in 1:3){
112         post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i], y=y)$z
113         post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i+1], y=y)$z
114     }
115 }
116
117 post_mean2<-post_mean[,c(2:10)]
118 post_var2<-post_var[,c(2:10)]
119 mean(post_mean2)
120 sd(post_mean2)
121 mean(post_var2)
122 sd(post_var2)
123 y<-rnorm(100,mean=100,sd=5)
124
125 post_mean<-matrix(NA,nrow=3,ncol=100)
126 post_var<-matrix(NA,nrow=3,ncol=100)
127
128 post_mean[,1]<-c(3,4,5)
129 post_var[,1]<-c(3,4,5)
130
131 a=.001
132 b=.001
133 mu<-0
134 sigma<-100
135
136 for(i in 1:99){
137     for(x in 1:3){
138         post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i], y=y)$z
139         post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i+1], y=y)$z
140     }
141 }
142
143 post_mean2<-post_mean[,c(11:100)]
144 post_var2<-post_var[,c(11:100)]
145 mean(post_mean2)
146 sd(post_mean2)
147 mean(post_var2)
148 sd(post_var2)
149 y<-rnorm(1000,mean=100,sd=5)
150
151 post_mean<-matrix(NA,nrow=3,ncol=1000)
152 post_var<-matrix(NA,nrow=3,ncol=1000)
153
154 post_mean[,1]<-c(3,4,5)
155 post_var[,1]<-c(3,4,5)
156
157 a=.001

```

```

158 b=.001
159 mu<-0
160 sigma<-100
161
162 for(i in 1:999){
163   for(x in 1:3){
164     post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i], y=y)$z
165     post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i+1], y=y)$z
166   }
167 }
168
169 post_mean2<-post_mean[,c(101:1000)]
170 post_var2<-post_var[,c(101:1000)]
171 mean(post_mean2)
172 sd(post_mean2)
173 mean(post_var2)
174 sd(post_var2)
175 y<-rnorm(10000,mean=100,sd=5)
176
177 post_mean<-matrix(NA,nrow=3,ncol=1000)
178 post_var<-matrix(NA,nrow=3,ncol=1000)
179
180 post_mean[,1]<-c(3,4,5)
181 post_var[,1]<-c(3,4,5)
182
183 a=.001
184 b=.001
185 mu<-0
186 sigma<-100
187
188 for(i in 1:999){
189   for(x in 1:3){
190     post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i], y=y)$z
191     post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i+1], y=y)$z
192   }
193 }
194
195 post_mean2<-post_mean[,c(101:1000)]
196 post_var2<-post_var[,c(101:1000)]
197 mean(post_mean2)
198 sd(post_mean2)
199 mean(post_var2)
200 sd(post_var2)
201 set.seed(101)
202 y<-rnorm(100,mean=100,sd=5)
203
204 post_mean<-matrix(NA,nrow=3,ncol=100)
205 post_var<-matrix(NA,nrow=3,ncol=100)
206
207 post_mean[,1]<-c(3,4,5)
208 post_var[,1]<-c(3,4,5)
209
210 a=.001

```

```

211 b=.001
212 mu<-0
213 sigma<-100
214
215 for(i in 1:99){
216   for(x in 1:3){
217     post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i], y=y)$z
218     post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i+1], y=y)$z
219   }
220 }
221
222 post_mean2<-post_mean[,c(11:100)]
223 post_var2<-post_var[,c(11:100)]
224 mean(post_mean2)
225 sd(post_mean2)
226 mean(post_var2)
227 sd(post_var2)
228 set.seed(10)
229 y<-rnorm(100,mean=100,sd=5)
230
231 post_mean<-matrix(NA,nrow=3,ncol=10000)
232 post_var<-matrix(NA,nrow=3,ncol=10000)
233
234 post_mean[,1]<-c(3,4,5)
235 post_var[,1]<-c(3,4,5)
236
237 a=.001
238 b=.001
239 mu<-0
240 sigma<-100
241
242 for(i in 1:9999){
243   for(x in 1:3){
244     post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i], y=y)$z
245     post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i+1], y=y)$z
246   }
247 }
248
249 post_mean2<-post_mean[,c(2:10000)]
250 post_var2<-post_var[,c(2:10000)]
251
252 # mean traceplot
253 df_mean = data.frame(iteration=rep(1:10000, times=3),
254                       chain=as.character(rep(1:3, each=10000)),
255                       mean=c(post_mean[1,], post_mean[2,], post_mean[3,]))
256
257 # mean traceplot
258 ggplot(df_mean %>% filter(iteration>1))+
259   geom_line(aes(x=iteration, y=mean, group=chain, color=chain), alpha=0.6)+
260   labs(title="Posterior Mean Trace Plot", x="Iteration", y="Mean")
261 #variance traceplot
262 df_var = data.frame(iteration=rep(1:10000, times=3),
263                     chain=as.character(rep(1:3, each=10000)),

```

```

264         var=c(post_var[1,], post_var[2,], post_var[3,]))
265
266     # variance traceplot
267     ggplot(df_var %>% filter(iteration>1))+
268       geom_line(aes(x=iteration, y=var, group=chain, color=chain), alpha=0.6)+
269       labs(title="Posterior Variance Trace Plot", x="Iteration", y="Variance")
270
271     mean(allmean)
272     sd(allmean)
273
274     mean(allvar)
275     sd(allvar)
276     set.seed(10)
277     y<-rnorm(100,mean=100,sd=5)
278     post_mean<-matrix(NA,nrow=3,ncol=10000)
279     post_var<-matrix(NA,nrow=3,ncol=10000)
280
281     post_mean[,1]<-c(3,4,5)
282     post_var[,1]<-c(3,4,5)
283
284     a=.001
285     b=.001
286     mu<-0
287     sigma<-100
288
289     for(i in 1:9999){
290       for(x in 1:3){
291
292         post_var[x,i+1]<-draw_var(alpha_0=a, beta_0=b, theta=post_mean[x,i], y=y)$z
293         post_mean[x,i+1]<-draw_mean(mu_0=mu, sigma.sq_0=sigma, varsigma.sq=post_var[x,i+1], y=y)$z
294       }
295     }
296
297     post_mean2<-post_mean[,c(2:10000)]
298     post_var2<-post_var[,c(2:10000)]
299
300     mean(allmean)
301     sd(allmean)
302
303     mean(allvar)
304     sd(allvar)
305     # this R markdown chunk generates a code appendix

```