Yichen Dong HW 6

Yichen Dong

October 10, 2018

Problem 1

a. $g(\cdot|x^{(t)}) = f(\cdot|x^{(t)})$

$$r = \frac{f(x^0)\, g(x^{(t)}|x^0)}{f(x^{(t)})\, g(x^0|x^{(t)})}$$

Since $g(\cdot|x^{(t)}) = f(\cdot|x^{(t)})$

$$r = \frac{f(x^0)\, f(x^{(t)}|x^0)}{f(x^{(t)})\, f(x^0|x^{(t)})}$$

$$r = \frac{f(x^0)\, f_{x^{(t)},x^0}(x^{(t)},x^0)}{f_{x^0}(x^0)} \cdot \frac{f_{x^{(t)},x^0}(x^{(t)},x^0)}{f(x^{(t)})\, f_{x^{(t)}}(x^{(t)})}$$

cancel

$$r = 1$$

b. Show

$$f(x_1, x_2) = \frac{f_{x_1|x_2}(x_1|x_2)}{\int \frac{f_{x_1|x_2}(x_1|x_2)}{f_{x_2|x_1}(x_2|x_1)}\, dx_1}$$

$$= \frac{\frac{f(x_1, x_2)}{f_{x_2}(x_2)}}{\int \frac{\frac{f(x_1, y)}{f_{x_2}(x_2)}}{\frac{f(x_1, y)}{f_{x_1}(x_1)}} \cdot f_{x_1}(x_1)\, dx_1}$$

$$= \frac{\frac{f(x_1, x_2)}{f_{x_2}(x_2)}}{\int \frac{f_{x_1}(x_1)}{f_{x_2}(x_2)}\, dx_1}$$

Since $\int f_{x_1}(x_1)\, dx_1 = 1$,

$$= \frac{f(x_1, x_2)}{f_{x_2}(x_2)} \cdot f_{x_2}(x_2) = f(x_1, x_2)$$

a. bivariate normal:

$$f(x,y) = ce^{-\frac{1}{2}Q(x,y)}, \quad c = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}$$

$$Q(x,y) = (1-\rho^2)^{-1}\left[\left(\frac{x-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x-\mu_1}{\sigma_1}\right)\left(\frac{y-\mu_2}{\sigma_2}\right) + \left(\frac{y-\mu_2}{\sigma_2}\right)^2\right]$$

$Q(x,y)$ can be rewritten as

$$Q(x,y) = \frac{1}{(1-\rho^2)}\left[\left(\frac{x-\mu_1}{\sigma_1} - \rho\frac{y-\mu_2}{\sigma_2}\right)^2 + (1-\rho^2)\left(\frac{y-\mu_2}{\sigma_2}\right)^2\right]$$

$$= \frac{(x-a)^2}{(1-\rho^2)\sigma_1^2} + \frac{(y-\mu_2)^2}{\sigma_2^2} \quad \text{where} \quad a = \mu_1 + \rho\frac{\sigma_1}{\sigma_2}(y-\mu_2)$$

Then we can find the marginal distribution

$$f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\,dx = ce^{-\frac{(y-\mu_2)^2}{2\sigma_2^2}} \times \underbrace{\int_{-\infty}^{\infty} e^{-\frac{(x-a)^2}{2(1-\rho^2)\sigma_1^2}}}_{= \sqrt{2\pi}\cdot\sigma_1\cdot\sqrt{1-\rho^2}}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_2}e^{-\frac{(y-\mu_2)^2}{2\sigma_2^2}}$$

Similarly, we can set

$$Q(x,y) = \frac{1}{(1-\rho^2)}\left[\left(\frac{y-\mu_2}{\sigma_2} - \rho\frac{x-\mu_1}{\sigma_1}\right)^2 + (1-\rho^2)\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right]$$

to get $f_X(x) = \frac{1}{\sqrt{2\pi}\sigma_1}e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}$

Thus, if we replace x with x_1 and y with x_2, we get

$$[X_1 | X_2] = \frac{f_{X_1,X_2}(x_1,x_2)}{f_{X_2}(x_2)} = \frac{ce^{-\frac{1}{2}Q(x_1,x_2)}}{\frac{1}{\sqrt{2\pi}\sigma_2}\cdot e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}}}$$

$$= \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \cdot e^{-\frac{1}{2}\left(\frac{(x_1-a)^2}{(1-\rho^2)\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2}\right)}$$

$$\frac{1}{\sqrt{2\pi}\,\sigma_2} \cdot e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}}$$

$$= \frac{1}{\sqrt{2\pi(1-\rho^2)}\cdot\sigma_1} \cdot e^{-\frac{(x_1-a)^2}{2(1-\rho^2)\sigma_1^2} - \frac{(x_2-\mu_2)^2}{2\sigma^2} + \frac{(x_2-\mu_2)^2}{2\sigma^2}}$$

$$= \frac{1}{\sqrt{2\pi(1-\rho^2)}\cdot\sigma_1} \cdot e^{-\frac{(x_1-a)^2}{2(1-\rho^2)\sigma_1^2}}$$

And thus we can see

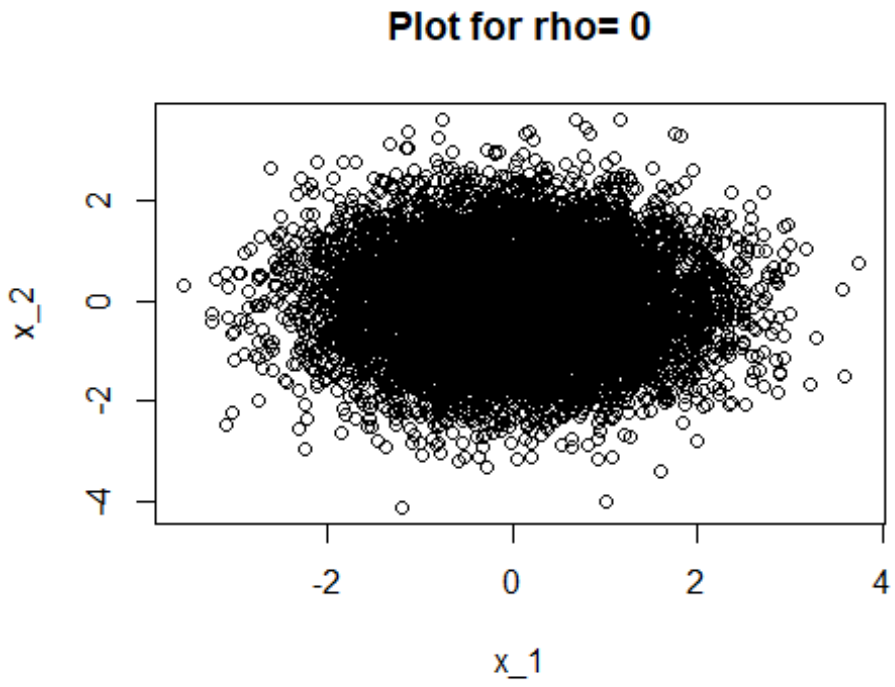
$$[x_1 \mid x_2] \sim N\left(\overbrace{\mu_1 + \rho\frac{\sigma_1}{\sigma_2}(x_2-\mu_2)}^{a},\ \sigma_1^2\cdot(1-\rho^2)\right)$$

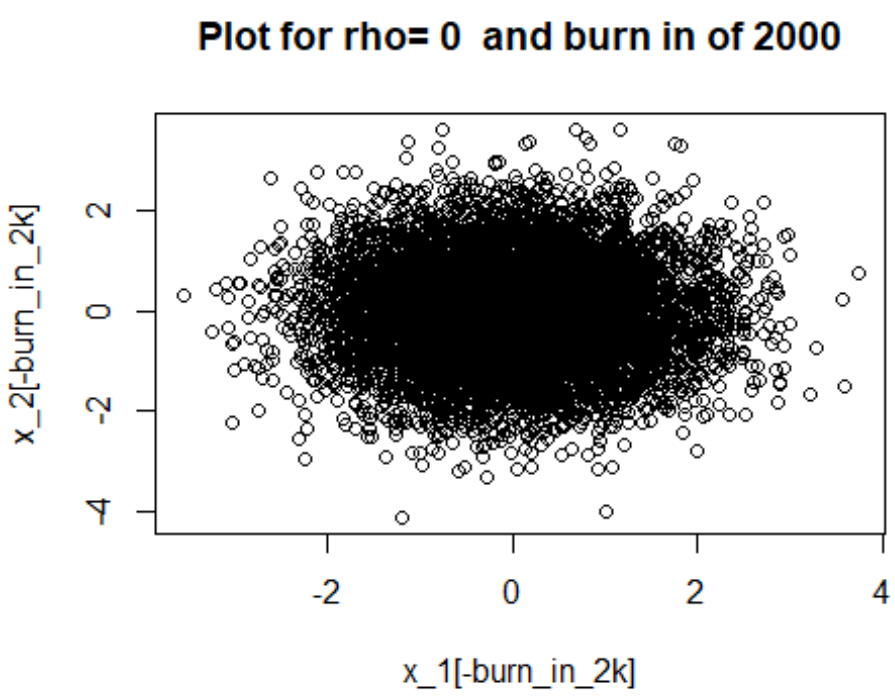
A similar thing can be shown for

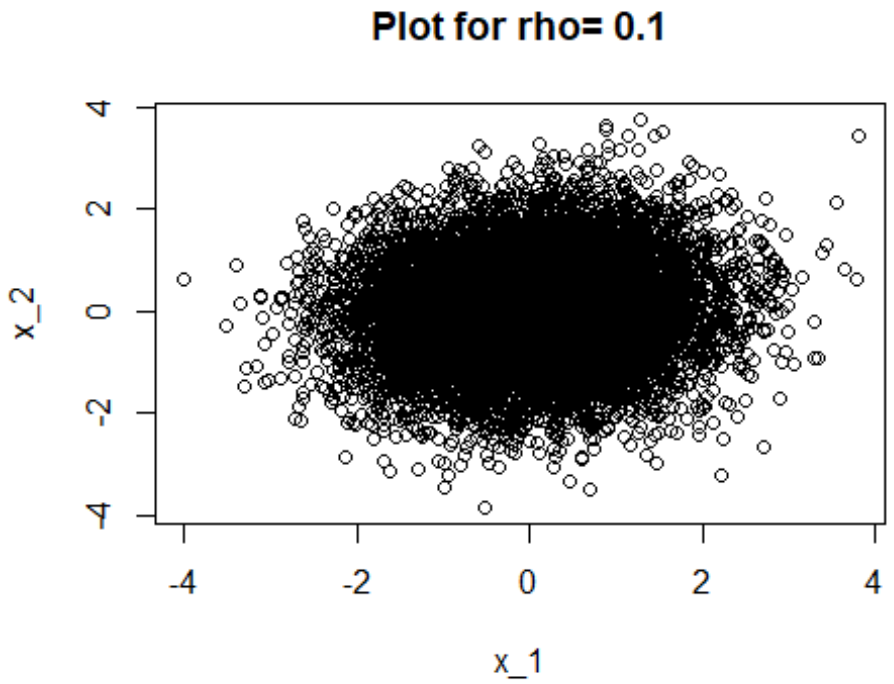
$$[x_2 \mid x_1) \sim N\left(\mu_2 + \rho\frac{\sigma_2}{\sigma_1}(x_1-\mu_1),\ \sigma_2^2(1-\rho^2)\right)$$

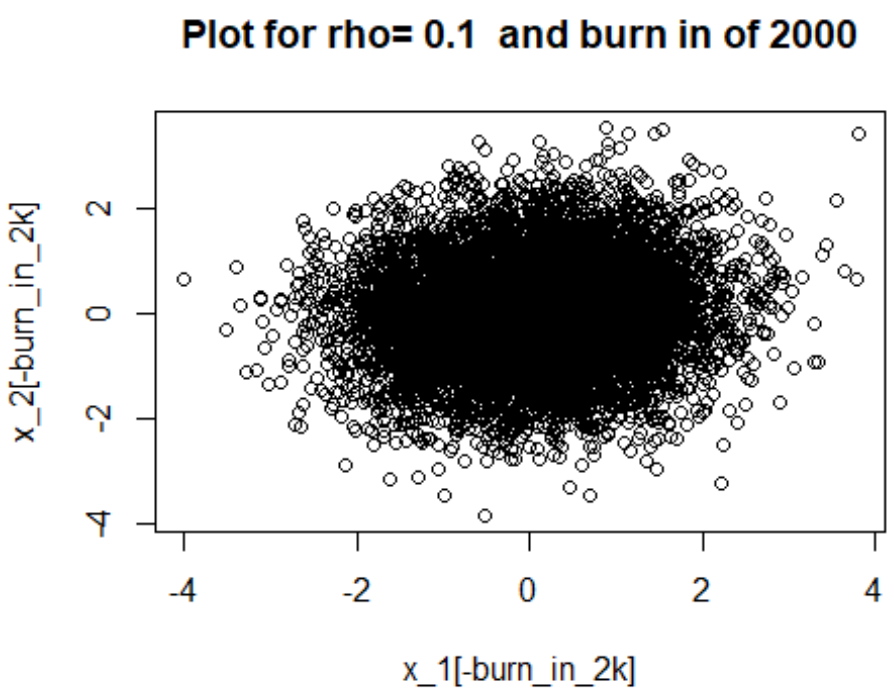
Problem 2 Part 2

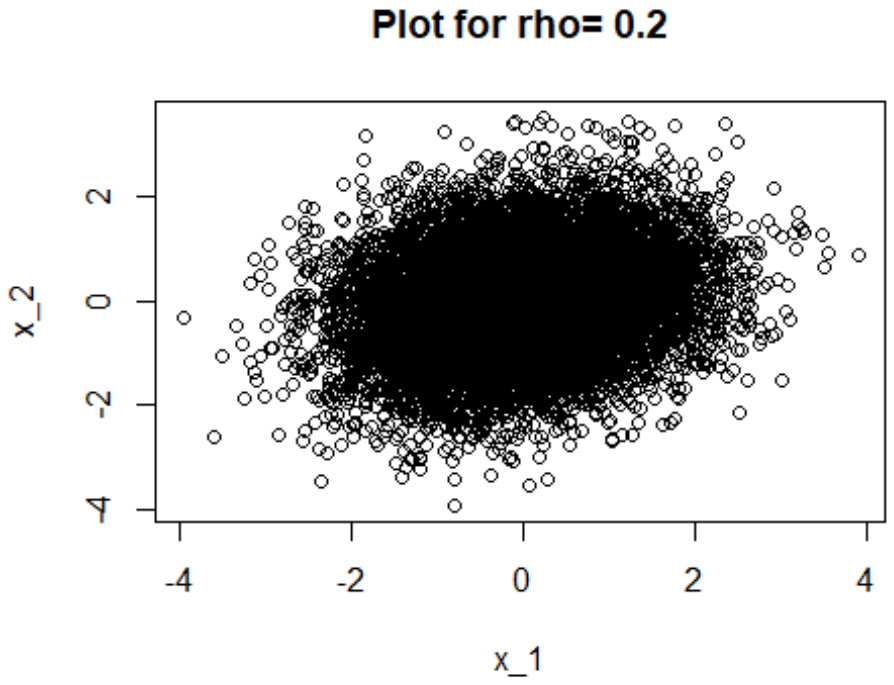
```
set.seed(1234)
summary = data.frame("mean_x_1" = numeric(),"var_x_1" = numeric(), "mean_x_2"
= numeric(),"var_x_2" = numeric(), "corr" =numeric(), "rho" = numeric())
counter = 1
burn_in_2k = 1:2000
for(rho in c(0,.1,.2,.3,.4,.5)){
  itr = 10000
  u_1 = 0
  u_2 = 0
  s_1 = 1
  s_2 = 1
  x_1 = NULL
  x_2 = NULL
  x_1[1] = 0
  x_2[1] = 0

  for(i in 1:itr){
    x_1[i+1] = rnorm(1,u_1+rho*s_1/s_2*(x_2[i]-u_2),sqrt(s_1^2*(1-rho^2)))
    x_2[i+1] = rnorm(1,u_2+rho*s_2/s_1*(x_1[i+1]-u_1),sqrt(s_2^2*(1-rho^2)))
  }

  plot(x_1,x_2)
  title(paste("Plot for rho=", rho))
  plot(x_1[-burn_in_2k],x_2[-burn_in_2k])
  title(paste("Plot for rho=", rho, " and burn in of 2000"))

  summary[counter,1] = mean(x_1)
  summary[counter,2]=var(x_1)
  summary[counter,3]=mean(x_2)
  summary[counter,4]=var(x_2)
  summary[counter,5]=cor(x_1,x_2)
  summary[counter,6]=rho
  counter = counter + 1
}
```

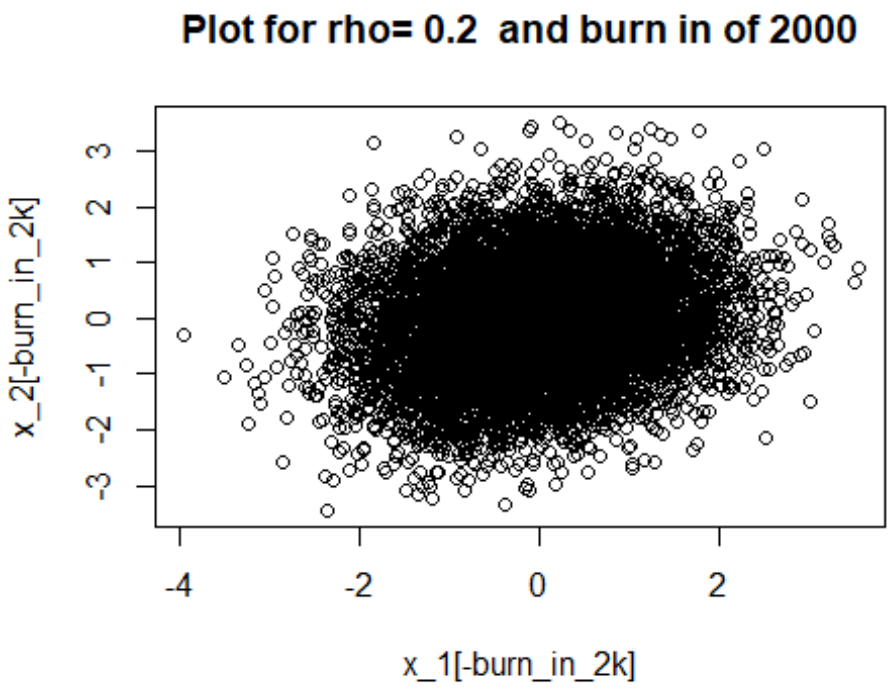

Plot for rho= 0
x_2
x_1
-4
-2
0
2
-2
0
2
4

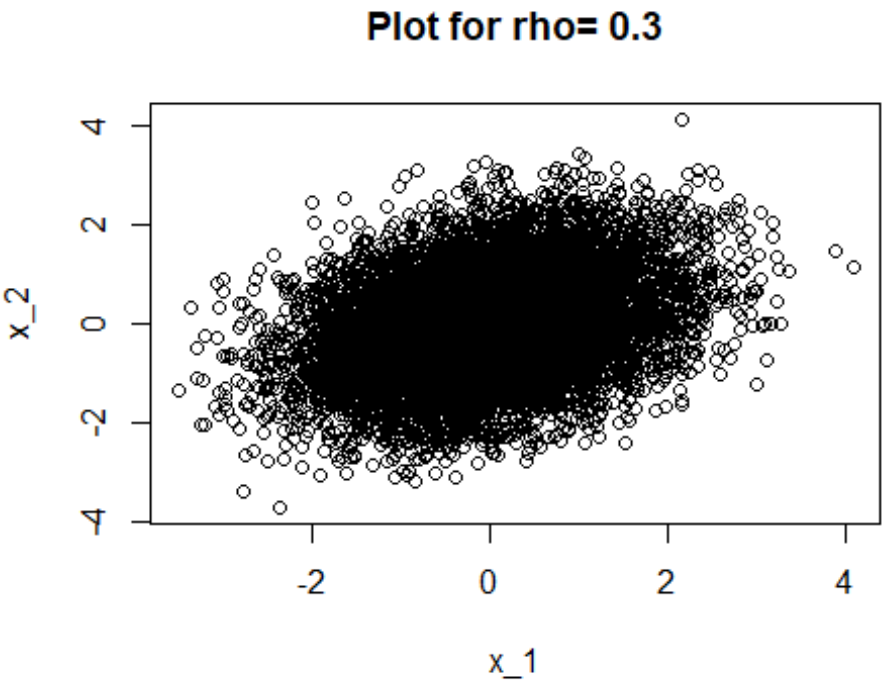

Plot for rho= 0 and burn in of 2000
x_2[-burn_in_2k]
x_1[-burn_in_2k]
-4
-2
0
2
-2
0
2
4

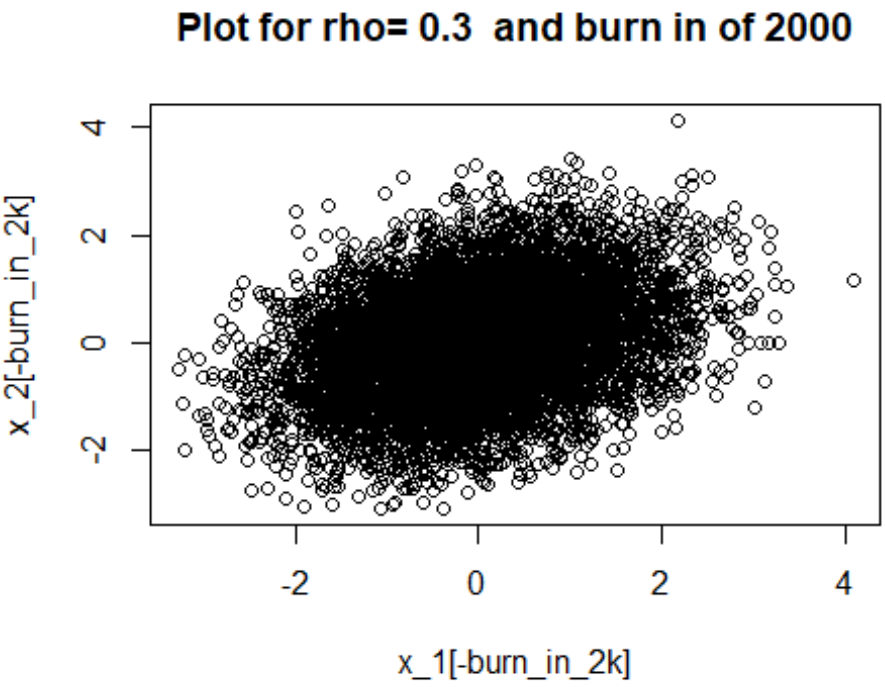

Plot for rho= 0.1
x_2
x_1
4
2
0
-2
-4
-4
-2
0
2
4

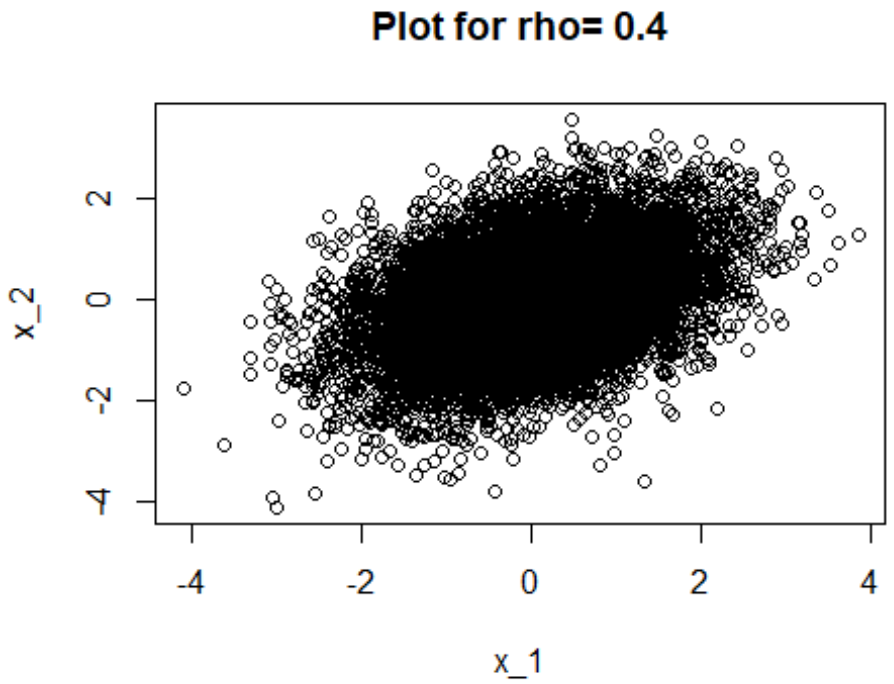

Plot for rho= 0.1 and burn in of 2000
x_2[-burn_in_2k]
x_1[-burn_in_2k]
2
0
-2
-4
-4
-2
0
2
4

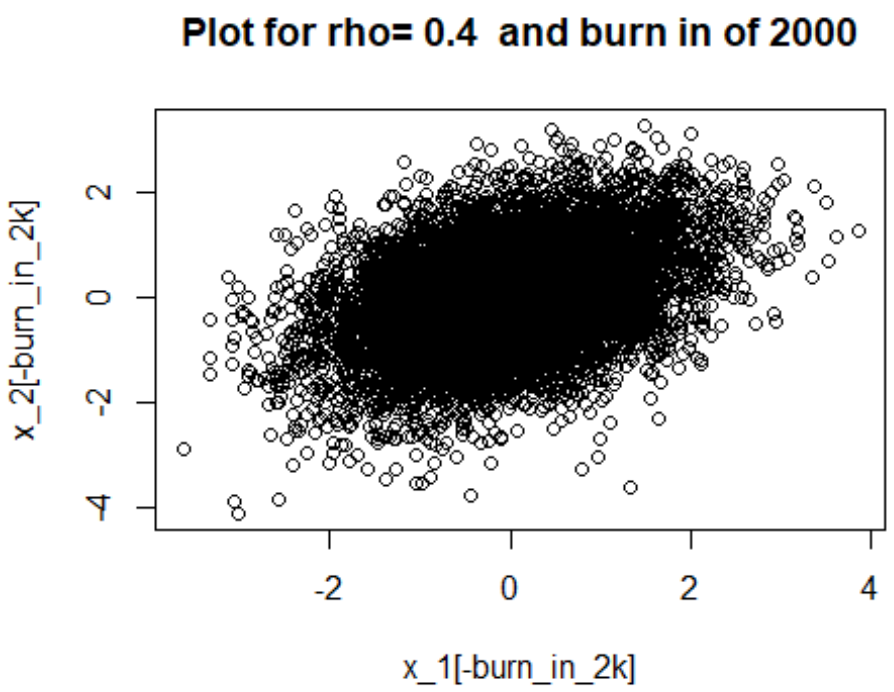

Plot for rho= 0.2
x_2
x_1
-4
-2
0
2
4

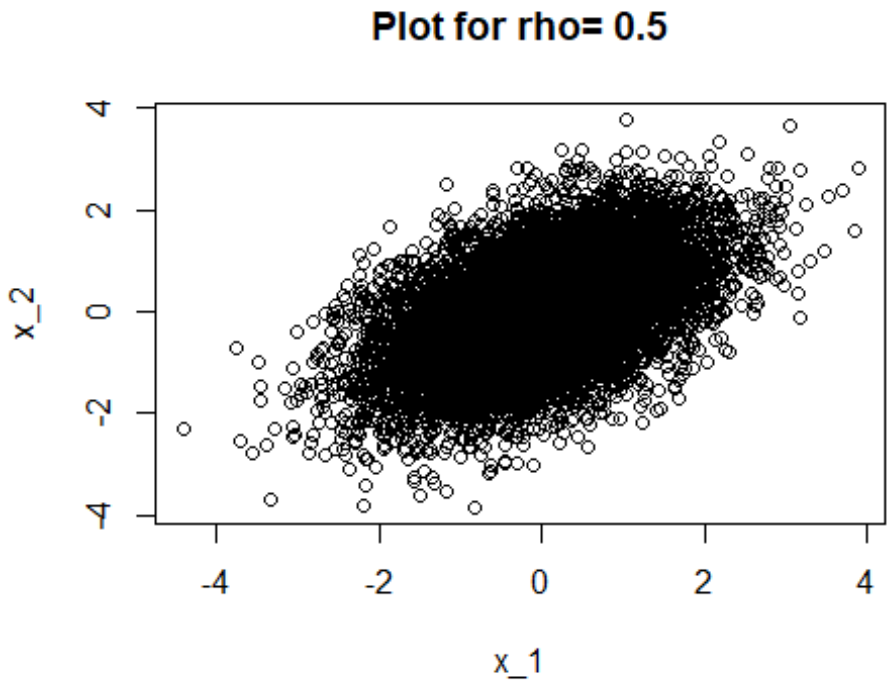

Plot for rho= 0.2 and burn in of 2000
x_2[-burn_in_2k]
x_1[-burn_in_2k]
-3
-2
-1
0
1
2
3
-4
-2
0
2

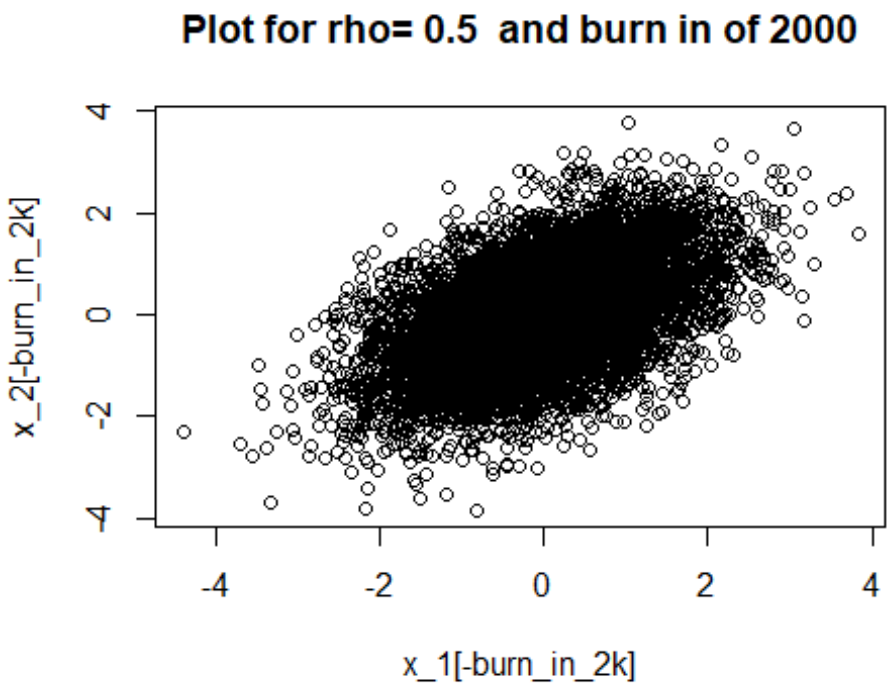

Plot for rho= 0.3
x_2
x_1
-4
-2
0
2
4


Plot for rho= 0.3 and burn in of 2000
x_2[-burn_in_2k]
x_1[-burn_in_2k]
-2
0
2
4


Plot for rho= 0.4
x_2
x_1
-4
-2
0
2
4


Plot for rho= 0.4 and burn in of 2000
x_2[-burn_in_2k]
x_1[-burn_in_2k]
-4
-2
0
2
4





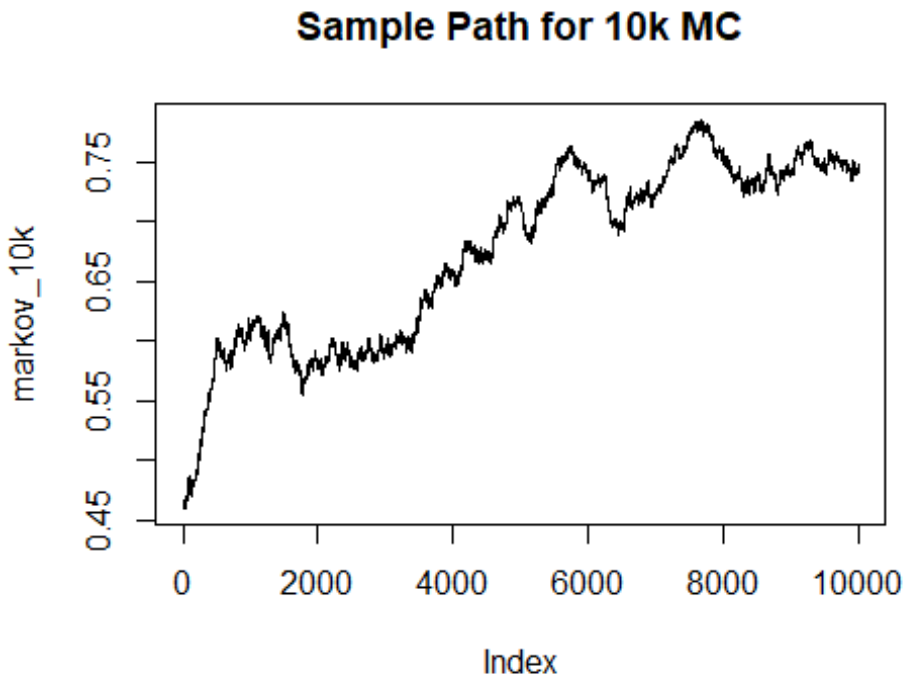

```
summary

##       mean_x_1   var_x_1     mean_x_2   var_x_2         corr rho
## 1  0.007790845 0.9932918 -0.009421811 0.9936796 -0.007484543 0.0
```

```
## 2  0.011437529 1.0218916  0.005248704 0.9834821  0.098044435 0.1
## 3 -0.006625190 1.0014226 -0.006211513 1.0042913  0.199493907 0.2
## 4 -0.008315305 1.0081808  0.010501223 1.0006561  0.311769035 0.3
## 5  0.019291907 1.0027863  0.018648848 0.9833818  0.392822570 0.4
## 6  0.013711359 0.9902929  0.001580508 0.9992635  0.499122714 0.5
```

We can see that the Gibbs sampling appears to be an appropriate method for this problem. The mean and variance of x_1 and x_2 were both very close to their actual values for all values of rho, and the correlation is also very close to the given correlation of rho.

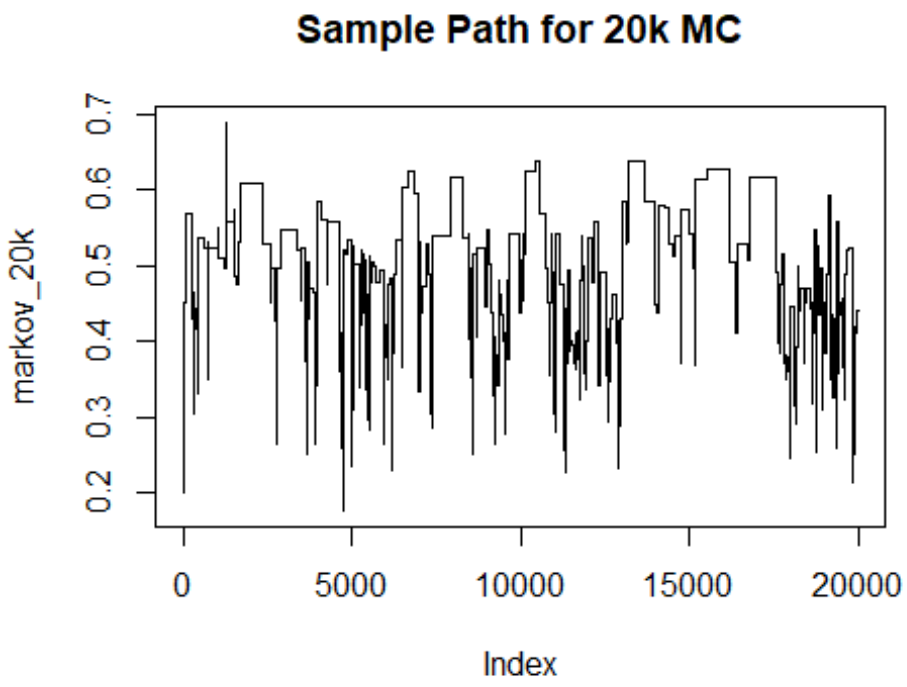
Problem 3

```
markov_10k = scan("Module 6 Data Sets/MCdata1.txt")
markov_20k = scan("Module 6 Data Sets/MCdata2.txt")

plot(markov_10k,type = "l")
title("Sample Path for 10k MC")
```



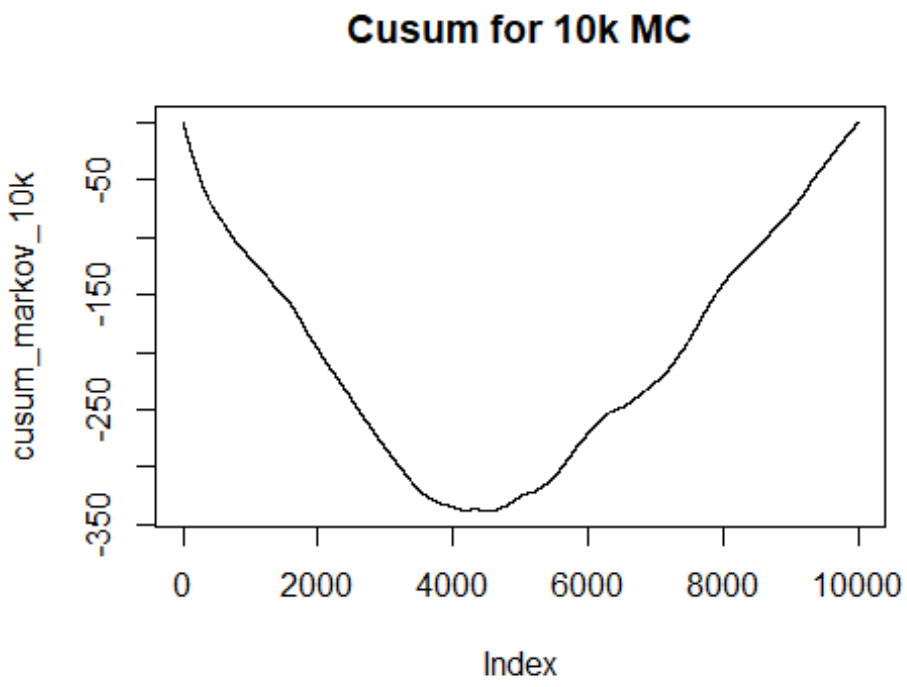
This does not look like it's wiggling vigorously about a single region

```
plot(markov_20k,type = "l")
title("Sample Path for 20k MC")
```



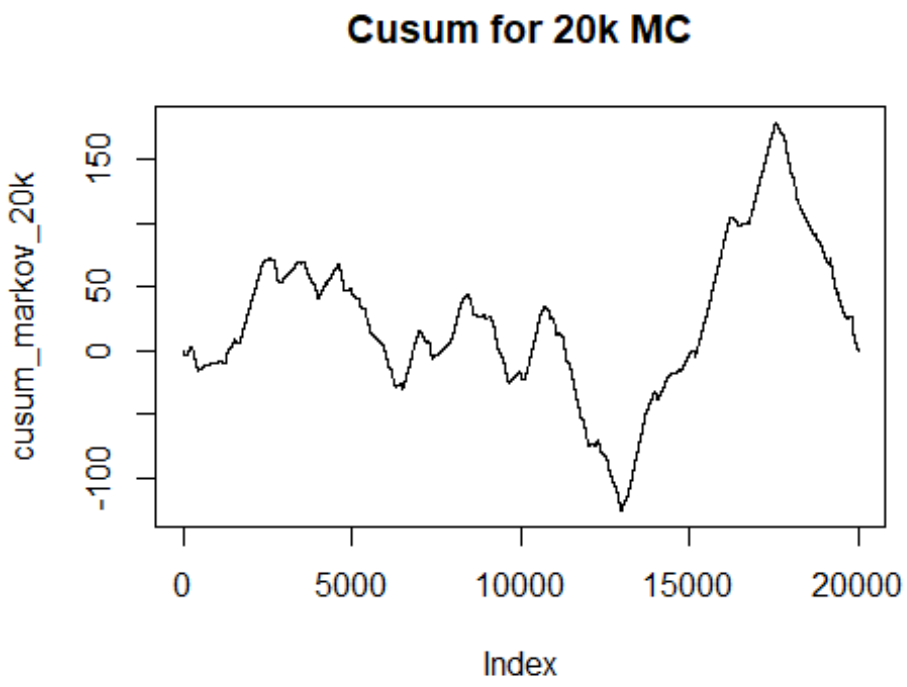
This looks like it's wiggling more around the same region, but I wouldn't exactly call it vigorous in some places.

```
theta_10k = mean(markov_10k)
theta_20k = mean(markov_20k)

cusum_markov_10k = NULL
for(i in 1:length(markov_10k)){
  cusum_markov_10k[i] = sum(markov_10k[1:i] - theta_10k)
}
plot(cusum_markov_10k, type = "l")
title("Cusum for 10k MC")
```



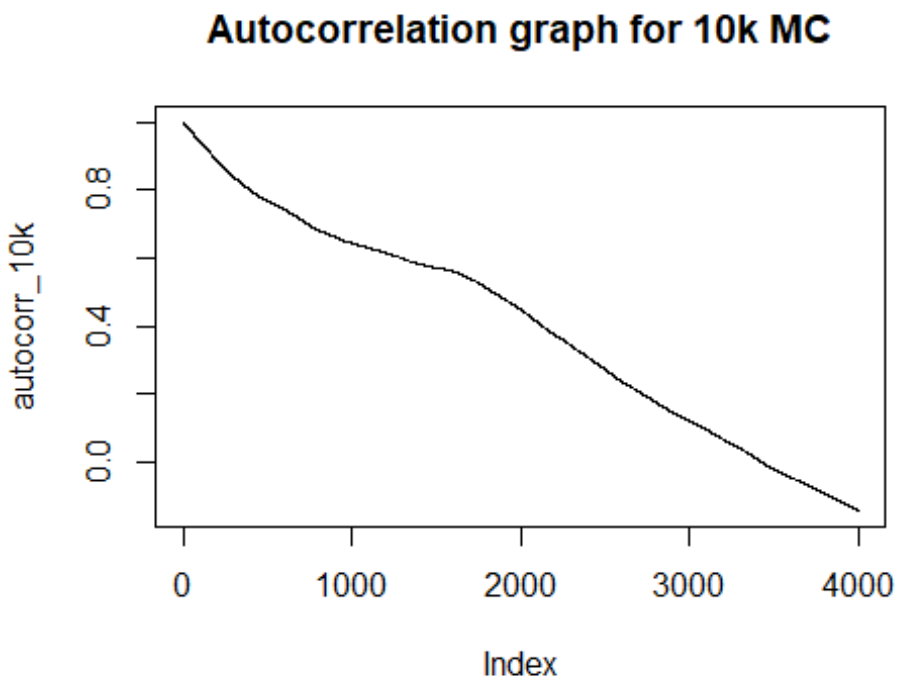
This isn't wiggly or only contain small excursions from 0. In fact, the excursion it makes is only negative until about the 5000 mark, then it's only positive!

```
cusum_markov_20k = NULL
for(i in 1:length(markov_20k)){
  cusum_markov_20k[i] = sum(markov_20k[1:i] - theta_20k)
}
plot(cusum_markov_20k, type = "l")
title("Cusum for 20k MC")
```



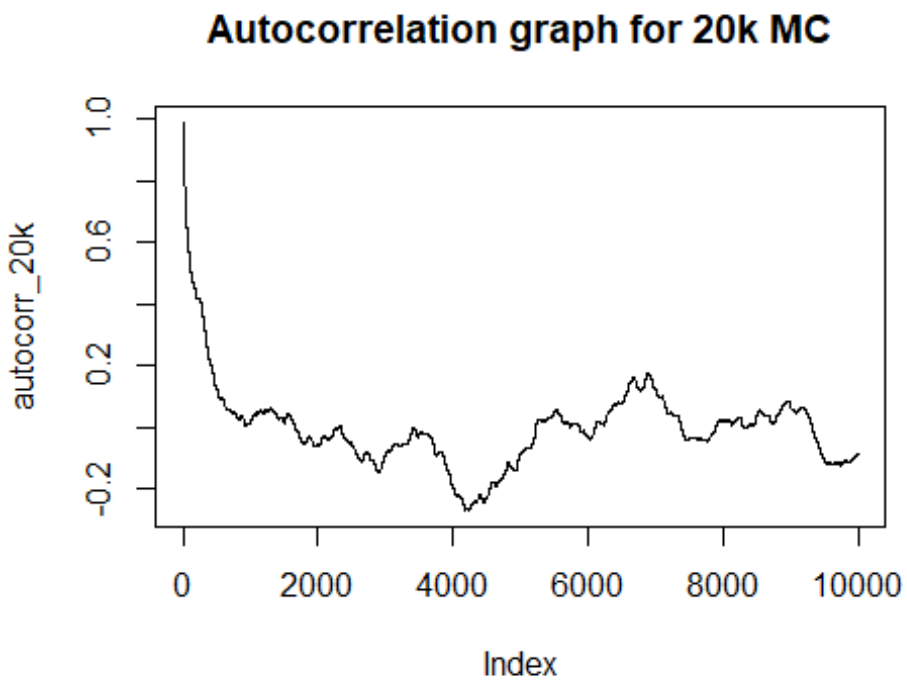
This looks like it's somewhat centered around 0 in the beginning, and has a sort of wiggly shape to it, but it starts losing its wiggliness after 10000 and has periods of going only down or up. It also starts getting larger deviations from 0.

```
autocorr_10k = NULL
for(i in 1:4000){
  R_i = NULL
  C_i = 0
  C_0 = 0
  x_bar = mean(markov_10k)
  n = length(markov_10k)
  for(t in 1:(length(markov_10k)-i)){
    C_i = C_i + 1/n*(markov_10k[t] - x_bar)*(markov_10k[t+i] - x_bar)
  }
  for(t in 1:length(markov_10k)){
    C_0 = C_0 + 1/n*(markov_10k[t] - x_bar)^2
  }
  R_i = C_i/C_0
  autocorr_10k[i] = R_i
}
plot(autocorr_10k,type = "l")
title("Autocorrelation graph for 10k MC")
```



While the correlation is going down linearly, I'm not sure if the pace of this is quick enough to say that this chain has good mixing properties. I think we want to see the autocorrelation decrease rapidly initially.

```
autocorr_20k = NULL
for(i in 1:10000){
  R_i = NULL
  C_i = 0
  C_0 = 0
  x_bar = mean(markov_20k)
  n = length(markov_20k)
  for(t in 1:(length(markov_20k)-i)){
    C_i = C_i + 1/n*(markov_20k[t] - x_bar)*(markov_20k[t+i] - x_bar)
  }
  for(t in 1:length(markov_20k)){
    C_0 = C_0 + 1/n*(markov_20k[t] - x_bar)^2
  }
  R_i = C_i/C_0
  autocorr_20k[i] = R_i
}

plot(autocorr_20k,type = "l")
title("Autocorrelation graph for 20k MC")
```



This autocorrelation graph looks pretty good initally, as there is a steep decline in correlation initally. However, it appears that around a lag of 1000 the correlation stops decreasing and starts going back up and then starts being correlated in the negative direction. It does seem to hang around 0, so I would say that the mixing properties are ok.

Based on these graphs, it seems that Markov Chain 1 has both decent mixing but very poor convergence. The correlation in the autocorrelation plot is going down linearly over time, but it doesn't have the sleep decline in the beginning. The sample path is wiggling vigorously, but not around a stable distribution. The cusum graph is not centered around 0 at all, and is not wiggly. Thus, convergence is pretty bad.

Markov Chain 2 appears to have good mixing initially, but eventually the mixing becomes worse past the 10000th mark. This can be seen in the later stages of the sample path, where it stays at the same value for large periods of time. The convergence also seems good initially, but in the later stages diverges from 0 in the Cusum plot.

Problem 4

```
multi_markov_1 = scan("Module 6 Data Sets/MultiMC1.txt")
multi_markov_2 = scan("Module 6 Data Sets/MultiMC2.txt")
multi_markov_3 = scan("Module 6 Data Sets/MultiMC3.txt")
multi_markov_4 = scan("Module 6 Data Sets/MultiMC4.txt")
multi_markov_5 = scan("Module 6 Data Sets/MultiMC5.txt")
multi_markov_6 = scan("Module 6 Data Sets/MultiMC6.txt")
multi_markov_7 = scan("Module 6 Data Sets/MultiMC7.txt")
multi_markov_all = cbind(multi_markov_1,multi_markov_2,multi_markov_3,multi_m
arkov_4,multi_markov_5,multi_markov_6,multi_markov_7)
```

```r
##Entire chain with D=0 and L=1000
D=0
L=1000
J=7
x_bar_j = NULL
for(i in 1:J){
  x_bar_j[i]=sum(multi_markov_all[(D+1):(D+L),i])/L
}

x_bar_dot = mean(x_bar_j)
B = L/(J-1)*sum((x_bar_j-x_bar_dot)^2)
s_j_squared = NULL
for(i in 1:J){
  s_j_squared[i] = 1/(L-1) * sum((multi_markov_all[(D+1):(D+L),i] - x_bar_j[i
])^2)
}
W = mean(s_j_squared)
R = (((L-1)/L)*W+(1/L)*B)/W
sqrt_R = sqrt(R)
paste("Sqrt_R for D =",D,"and L =",L,"is", round(sqrt_R,4), ",B is", round(B,
4), ",W is",round(W,4))
```

```
## [1] "Sqrt_R for D = 0 and L = 1000 is 1.0082 ,B is 0.1137 ,W is 0.0065"
```

```r
##Entire chain with D=500 and L=500
D=500
L=500
J=7
x_bar_j = NULL
for(i in 1:J){
  x_bar_j[i]=sum(multi_markov_all[(D+1):(D+L),i])/L
}

x_bar_dot = mean(x_bar_j)
B = L/(J-1)*sum((x_bar_j-x_bar_dot)^2)
s_j_squared = NULL
for(i in 1:J){
  s_j_squared[i] = 1/(L-1) * sum((multi_markov_all[(D+1):(D+L),i] - x_bar_j[i
])^2)
}
W = mean(s_j_squared)
R = (((L-1)/L)*W+(1/L)*B)/W
sqrt_R = sqrt(R)
paste("Sqrt_R for D =",D,"and L =",L,"is", round(sqrt_R,4), ",B is", round(B,
4), ",W is",round(W,4))
```

```
## [1] "Sqrt_R for D = 500 and L = 500 is 1.0212 ,B is 0.1083 ,W is 0.0048"
```

```r
##First 500 elements with D=0 and L=500
D=0
L=500
```

```r
J=7
x_bar_j = NULL
for(i in 1:J){
  x_bar_j[i]=sum(multi_markov_all[(D+1):(D+L),i])/L
}

x_bar_dot = mean(x_bar_j)
B = L/(J-1)*sum((x_bar_j-x_bar_dot)^2)
s_j_squared = NULL
for(i in 1:J){
  s_j_squared[i] = 1/(L-1) * sum((multi_markov_all[(D+1):(D+L),i] - x_bar_j[i
])^2)
}
W = mean(s_j_squared)
R = (((L-1)/L)*W+(1/L)*B)/W
sqrt_R = sqrt(R)
paste("Sqrt_R for D =",D,"and L =",L,"is", round(sqrt_R,4), ",B is", round(B,
4), ",W is",round(W,4))
```

```
## [1] "Sqrt_R for D = 0 and L = 500 is 1.0069 ,B is 0.0637 ,W is 0.008"
```

```r
##First 500 elements with D=250 and L=250
D=250
L=250
J=7
x_bar_j = NULL
for(i in 1:J){
  x_bar_j[i]=sum(multi_markov_all[(D+1):(D+L),i])/L
}

x_bar_dot = mean(x_bar_j)
B = L/(J-1)*sum((x_bar_j-x_bar_dot)^2)
s_j_squared = NULL
for(i in 1:J){
  s_j_squared[i] = 1/(L-1) * sum((multi_markov_all[(D+1):(D+L),i] - x_bar_j[i
])^2)
}
W = mean(s_j_squared)
R = (((L-1)/L)*W+(1/L)*B)/W
sqrt_R = sqrt(R)
paste("Sqrt_R for D =",D,"and L =",L,"is", round(sqrt_R,4), ",B is", round(B,
4), ",W is",round(W,4))
```

```
## [1] "Sqrt_R for D = 250 and L = 250 is 1.0192 ,B is 0.057 ,W is 0.0053"
```

```r
##First 50 elements with D=0 and L=50
D=0
L=50
J=7
x_bar_j = NULL
for(i in 1:J){
```

```
  x_bar_j[i]=sum(multi_markov_all[(D+1):(D+L),i])/L
}

x_bar_dot = mean(x_bar_j)
B = L/(J-1)*sum((x_bar_j-x_bar_dot)^2)
s_j_squared = NULL
for(i in 1:J){
  s_j_squared[i] = 1/(L-1) * sum((multi_markov_all[(D+1):(D+L),i] - x_bar_j[i
])^2)
}
W = mean(s_j_squared)
R = (((L-1)/L)*W+(1/L)*B)/W
sqrt_R = sqrt(R)
paste("Sqrt_R for D =",D,"and L =",L,"is", round(sqrt_R,4), ",B is", round(B,
4), ",W is",round(W,4))
```

```
## [1] "Sqrt_R for D = 0 and L = 50 is 1.0452 ,B is 0.1282 ,W is 0.0228"
```

```
##First 50 elements with D=25 and L=25
D=25
L=25
J=7
x_bar_j = NULL
for(i in 1:J){
  x_bar_j[i]=sum(multi_markov_all[(D+1):(D+L),i])/L
}

x_bar_dot = mean(x_bar_j)
B = L/(J-1)*sum((x_bar_j-x_bar_dot)^2)
s_j_squared = NULL
for(i in 1:J){
  s_j_squared[i] = 1/(L-1) * sum((multi_markov_all[(D+1):(D+L),i] - x_bar_j[i
])^2)
}
W = mean(s_j_squared)
R = (((L-1)/L)*W+(1/L)*B)/W
sqrt_R = sqrt(R)
paste("Sqrt_R for D =",D,"and L =",L,"is", round(sqrt_R,4), ",B is", round(B,
4), ",W is",round(W,4))
```

```
## [1] "Sqrt_R for D = 25 and L = 25 is 1.4162 ,B is 0.0618 ,W is 0.0024"
```

It seemed that for the most part, the sqrt(R) was close to 1, except for the last one with only 25 elements. It seems that 50 elements might be too small. However, it appears that the sqrt(r) is practically the same for L=500 and L=1000 when D=0. It also seems that between chain variance is the smallest for D=250 and L=250.