Atlas-PS 6

David Atlas

10/2/2018

Problem 1

a)

To show that the Metropolis-Hastings ratio will always be equal to 1 when $g(\cdot|x^{(t)}) = f(\cdot|x^{(t)})$, we first define the ratio

$$R(x^{(t)}, X^*) = \frac{f(X^*)g(x^{(t)}|X^*)}{f(x^{(t)})g(X^*|x^{(t)})}.$$

Note that $g(\cdot|x^{(t)}) = f(\cdot|x^{(t)}) \forall x^{(t)} \implies f(X^*|x^{(t)}) = g(X^*|x^{(t)})$ and $ f(X^{(t)} | X^) = g(X^{(t)} / X^)$. In the Metropolis-Hastings Ratio, f is not conditional on previous values of the simulation, as it is the target distribution that we want to converge on. Therefore, $f(X^*) = g(X^*|x^{(t)})$ and $f(X^{(t)}) = g(X^{(t)}|X^*) \forall x^{(t)}$.

Now, it is trivial to show that

$$R(x^{(t)}, X^*) = \frac{g(x^{(t)}|X^*)f(X^*)}{f(x^{(t)})g(X^*|x^{(t)})} = \frac{f(x^{(t)}|X^*)f(X^*)}{f(x^{(t)})f(X^*|x^{(t)})} = \frac{f(x^{(t)})f(X^*)}{f(x^{(t)})f(X^*)} = 1.$$

This is intuitive too, as if the proposal distribution is equal to the target distribution, any draw from the proposal distribution should be included in the chain, and this will lead to convergence on the target distribution.

b)

We start with the definition of the conditional distribution

$$f_{X_1|X_2}(x_1|x_2) = \frac{f(x_1, x_2)}{f_{X_2}(x_2)},$$

or that the conditional distribution is equal to the joint distribution divided by the marginal distribution of the conditioning variable.

This implies that

$$\frac{f_{X_1|X_2}(x_1|x_2)}{f_{X_2|X_1}(x_2|x_1)} = \frac{f(x_1, x_2)f_{X_1}(x_1)}{f_{X_2}(x_2)f(x_1, x_2)} = \frac{f_{X_1}(x_1)}{f_{X_2}(x_2)}.$$

Integrating both sides of the equation with respect to dx_1, we get

$$\int_{-\infty}^{\infty} \frac{f_{X_1|X_2}(x_1|x_2)}{f_{X_2|X_1}(x_2|x_1)} dx_1 = \int_{-\infty}^{\infty} \frac{f_{X_1}(x_1)}{f_{X_2}(x_2)} dx_1 = \frac{1}{f_{X_2}(x_2)}$$

because $\int_{-\infty}^{\infty} f_{X_1}(x_1)dx_1 = 1$, as f_{X_1} is a probability distribution.

We reference the equations above to show that

$$f_{X_2}(x_2) = \frac{f(x_1, x_2)}{f_{X_1|X_2}(x_1|x_2)}$$

implying

$$\frac{1}{\int_{-\infty}^{\infty} \frac{f_{X_1|X_2}(x_1|x_2)}{f_{X_2|X_1}(x_2|x_1)} dx_1} = f_{X_2}(x_2) = \frac{f(x_1, x_2)}{f_{X_1|X_2}(x_1|x_2)}.$$

Multiplying across, we can say that

$$f(x_1, x_2) = \frac{f_{X_1|X_2}(x_1|x_2)}{\int_{-\infty}^{\infty} \frac{f_{X_1|X_2}(x_1|x_2)}{f_{X_2|X_1}(x_2|x_1)} dx_1}.$$

Problem 2

a)

We begin with the bivariate normal distribution:

$$f_{X_1X_2}(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \mathrm{e}^{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2 \cdot\right]}$$

We note that the conditional density $f_{X_1|X_2}(x_1|x_2) = \frac{f_{X_1X_2}(x_1,x_2)}{f_{X_2}(x_2)}$. In this case, this means that

$$f_{X_1|X_2}(x_1|x_2) = \frac{\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \mathrm{e}^{-\frac{1}{2}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2 \cdot\right]}}{\frac{1}{\sqrt{2\pi\sigma_2^2}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x_2-\mu_2}{\sigma_2}^2\right)}}$$

We can rewrite the exponent of the exponential of the bivariate normal as follows:

$$-\frac{1}{2(1-\rho^2)}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \rho^2\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2 + (1-\rho^2)\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2.$$

Therefore, expanding our terms and cancelling out where possible, we can write the conditional density as

$$f_{X_1|X_2}(x_1|x_2) = \frac{1}{\sqrt{2\pi\sigma_1^2(1-\rho^2)}} \mathrm{e}^{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \rho^2\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right]}$$

Note that the above exponent of the exponential function can be rewritten:

$$\begin{aligned}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \rho^2\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2 &= \left(\left(\frac{x_1-\mu_1}{\sigma_1}\right) - \rho\left(\frac{x_2-\mu_2}{\sigma_2}\right)^2\right)^2 \\ &= \frac{1}{\sigma_1^2}\left((x_1-\mu_1) - \rho\frac{\sigma_1}{\sigma_2}(x_2-\mu_2)\right)^2\end{aligned}$$

such that the conditional probability can be written as

$$f_{X_1|X_2}(x_1|x_2) = \frac{1}{\sqrt{2\pi\sigma_1^2(1-\rho^2)}} \mathrm{e}^{-\frac{1}{2}\frac{\left(x_1-\mu_1-\rho\frac{\sigma_1}{\sigma_2}(x_2-\mu_2)\right)^2}{\sigma_1^2(1-\rho^2)}}.$$

Given the symmetric nature of the density, we can write

$$f_{X_2|X_1}(x_2|x_1) = \frac{1}{\sqrt{2\pi\sigma_2^2(1-\rho^2)}} e^{-\frac{1}{2}\frac{\left(x_2-\mu_2-\rho\frac{\sigma_2}{\sigma_1}(x_1-\mu_1)\right)^2}{\sigma_2^2(1-\rho^2)}}$$

Note that these are both normally distributed variables, so we can write that

$$X_1|X_2 \sim N(\mu_1 + \rho\frac{\sigma_2}{\sigma_1}(x_2-\mu_2), \sigma_1^2(1-\rho^2))$$
$$X_2|X_1 \sim N(\mu_2 + \rho\frac{\sigma_1}{\sigma_2}(x_1-\mu_1), \sigma_2^2(1-\rho^2))$$

b)

Next, we generate Gibbs samples to determine if these distributions can be simulated with that methodology. We assume the variables are standard normal, with varying correlations.

```
# Define the conditional distributions
one_given_two <- function(rho, x2) rnorm(n=1, mean=(rho * x2), sd=(1-rho^2))
two_given_one <- function(rho, x1) rnorm(n=1, mean=(rho * x1), sd=(1-rho^2))

# Generate 1000 samples from each
data_list <- lapply(seq(0, .5, .1), function(rho){
  x2 <- 0
  samples <- matrix(ncol=2, nrow=0)
  for(i in 1:5000){
    # Get the samples
    x1 <- one_given_two(rho, x2)
    x2 <- two_given_one(rho, x1)
    samples <- rbind(samples, c(x1, x2))
  }
  return(samples)
})
```

To determine whether or not Gibbs sampling is a viable method here, we check the estimated parameters against their true values. We note that the mean is well estimated in all the samples, so we'll examine only the covariance matrix for each of the iterations.

```
estimates <- lapply(seq(1, length(data_list)), function(i){
  data_ <- data_list[[i]]
  rho <- seq(0, .5, .1)[i]
  res <- lapply(seq(1000, nrow(data_) - 1, 1000), function(burnin){
    # Get the burnin data only
    data <- data_[burnin:nrow(data_), ]
    # Get the covariance matrix
    data.frame(cov(data))
  })
  res
})
```

(Note: Not printing the results here because it was many dataframes, but I took a look at the output manually.)

Inspecting the results, we see that the correlation between the two variables creates a problem. All of the parameters are estimates well, except the variances of the individual variables. Because the variables are correlated, they clearly have a tendency to exhibit smaller variances together.

When there is no correlation, the sampling works well. As the correlation grows, the parameter estimates for the variance and the correlation degrades. Gibbs Sampling may not work so well for estimating all of the parameters here.

Problem 3

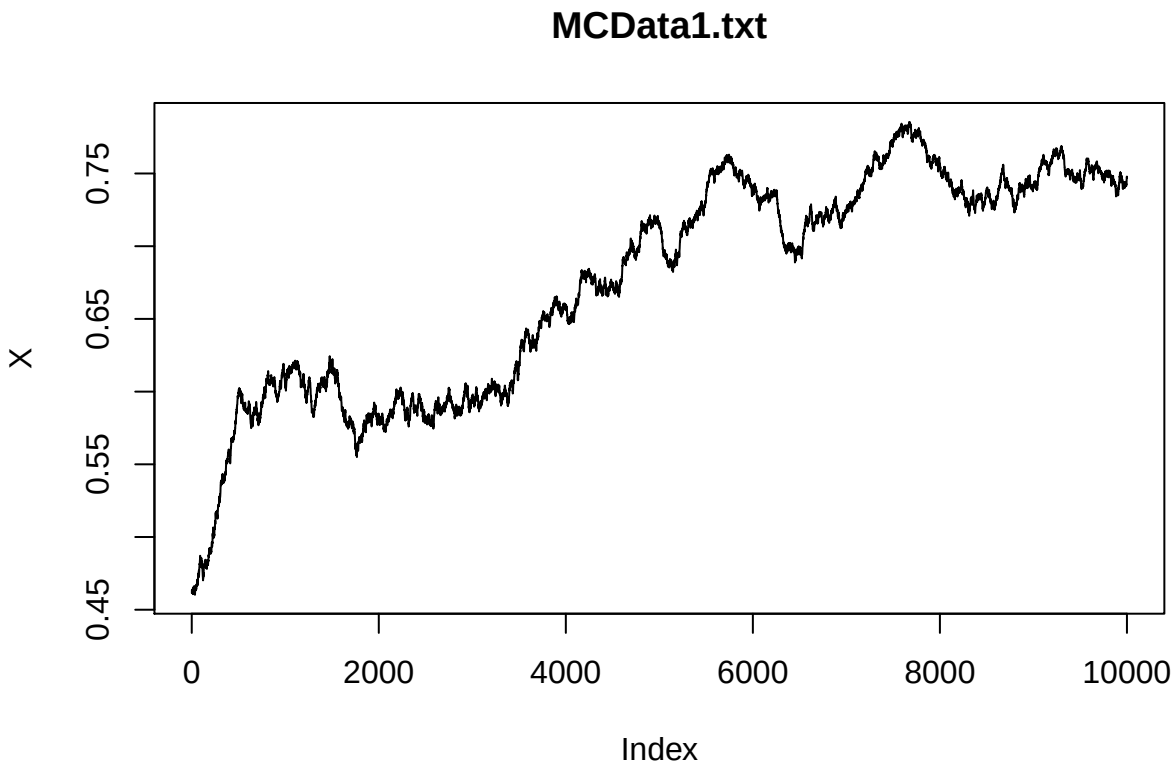
a) MCdata1.txt

We read in the first dataset from Blackboard.

```
mc_data1 <- scan("Module 6 Data Sets/MCdata1.txt")
```

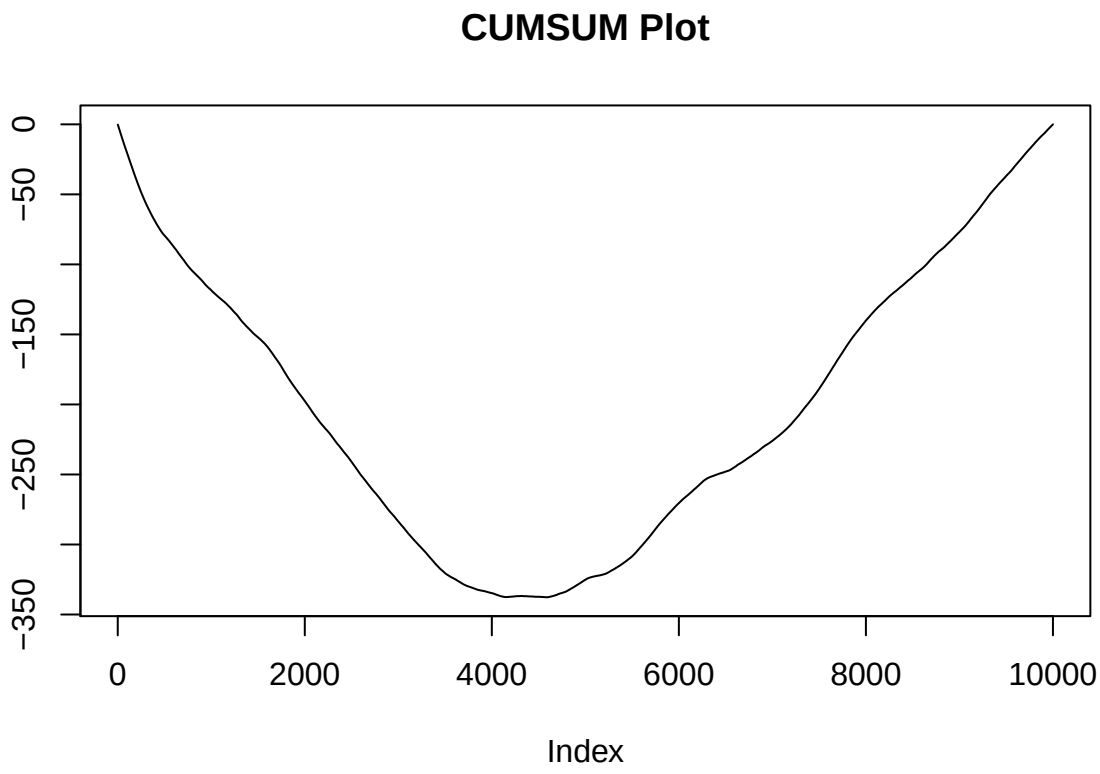
We plot the sample path for the first MC.

```
plot(mc_data1, type =  'l', main='MCData1.txt', ylab='X')
```



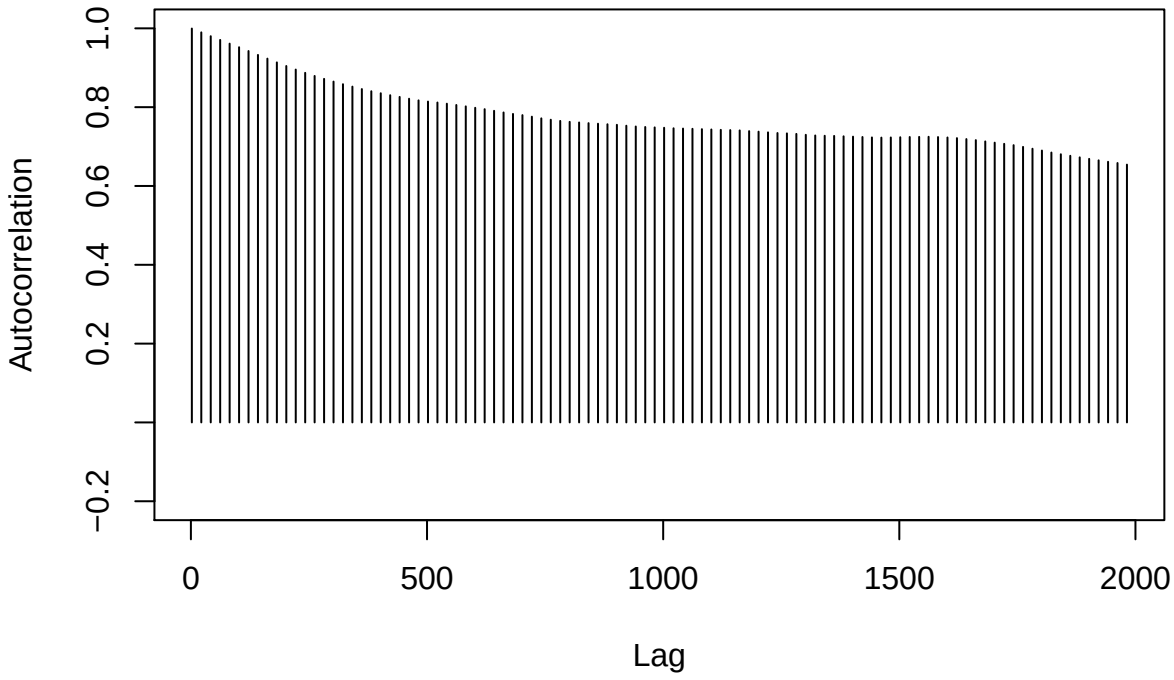
We plot the CUMSUM diagnostic plot next.

```
mu <- mean(mc_data1)
plot(cumsum(mc_data1 - mu), type='l', main='CUMSUM Plot', ylab='')
```



Next, we plot the autocorrelation plot for the chain.

```
autocorr <- function(data, lag){
  ct <- cov(data[1: (length(data) - lag + 1)], data[lag :length(data)])
  c0 <- var(data[1: (length(data) - lag + 1)])
  return(ct / c0)
}
n <- 2000
plot(seq(2, n, 20), sapply(seq(2, n, 20), function(lag) autocorr(mc_data1, lag)),
     type='h', ylim=c(-.2, 1), xlab="Lag", ylab="Autocorrelation")
```



Combining all three of the diagnostic plots, it seems as though the chain is not mixing well. The first plot shows the chain getting stuck in varies regions, but not bouncing around a specified region (the range of the stationary distribution).

The second plot does not show a CUMSUM that is wiggly with small excursions from zero. It's pretty clear that even if we included a burnin period, this would still be the case.

The third plot shows that the autocorrelation decays extremely slowly. Above, we show 2000 lags, and the autocorrelation is pretty strong throughout.

It seems as though this chain isn't mixing well, and is not converging at all.

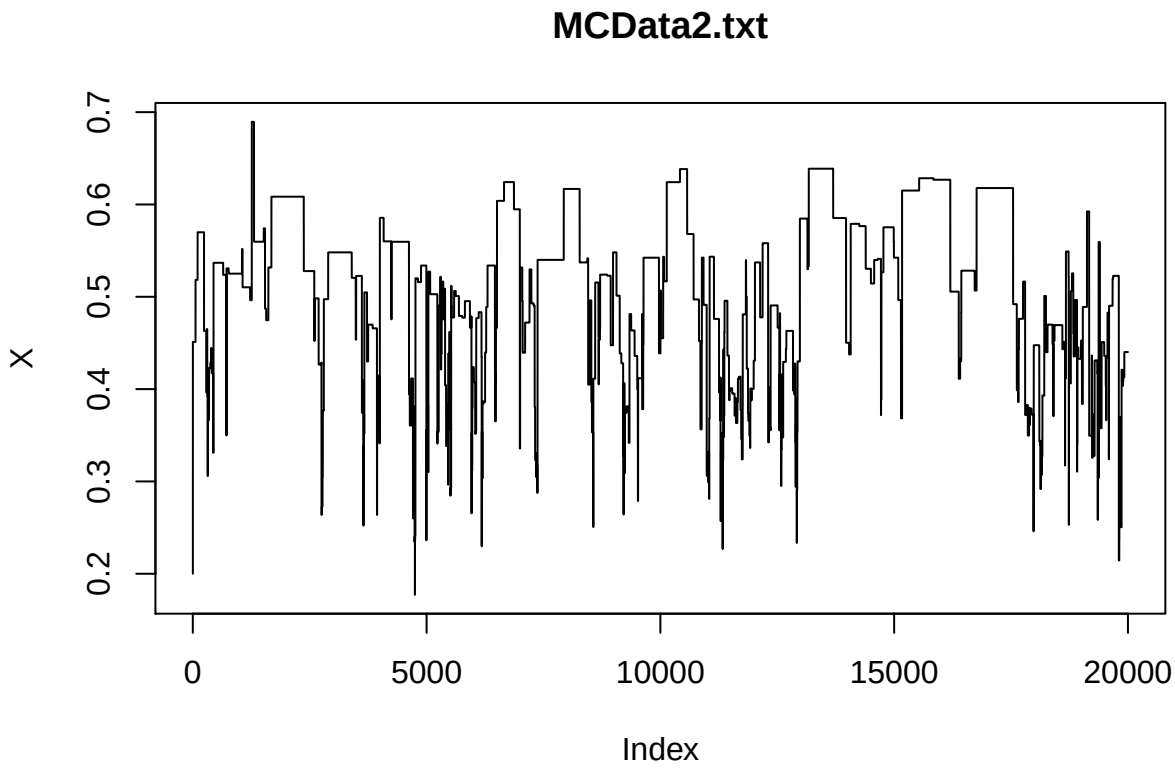
b) MCdata2.txt

We read in the second dataset from Blackboard.

```
mc_data2 <- scan("Module 6 Data Sets/MCdata2.txt")
```

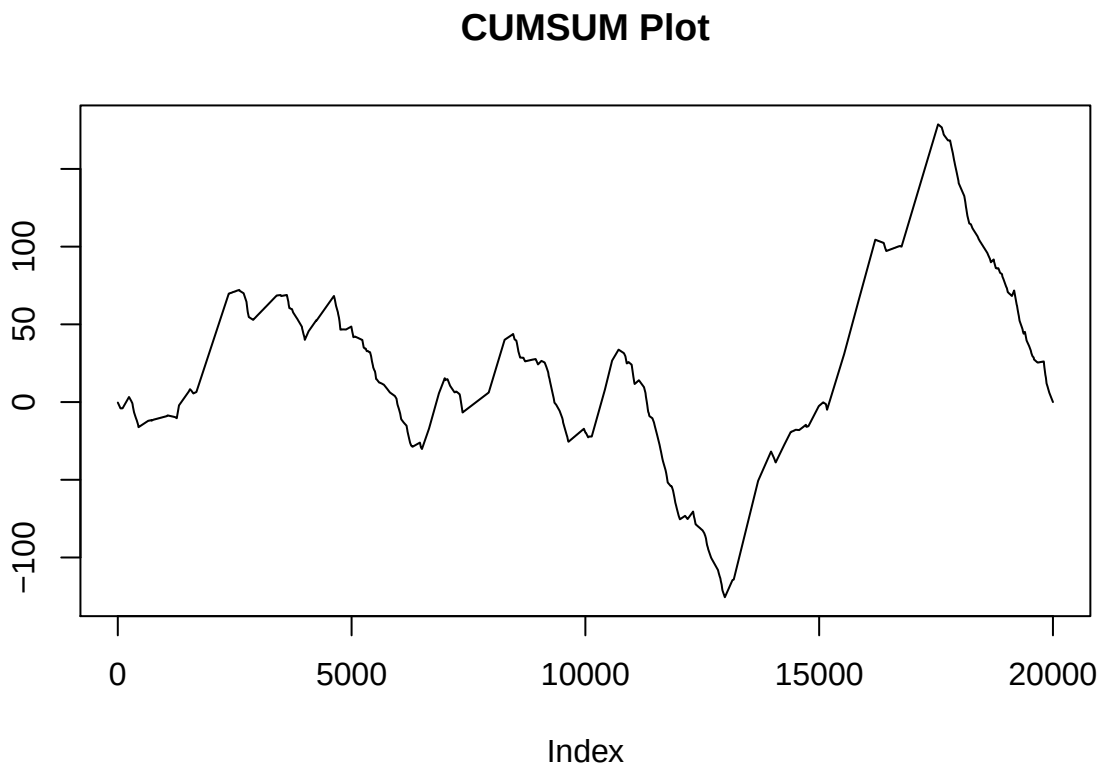
We plot the sample path for the first MC.

```
plot(mc_data2, type =  'l', main='MCData2.txt', ylab='X')
```



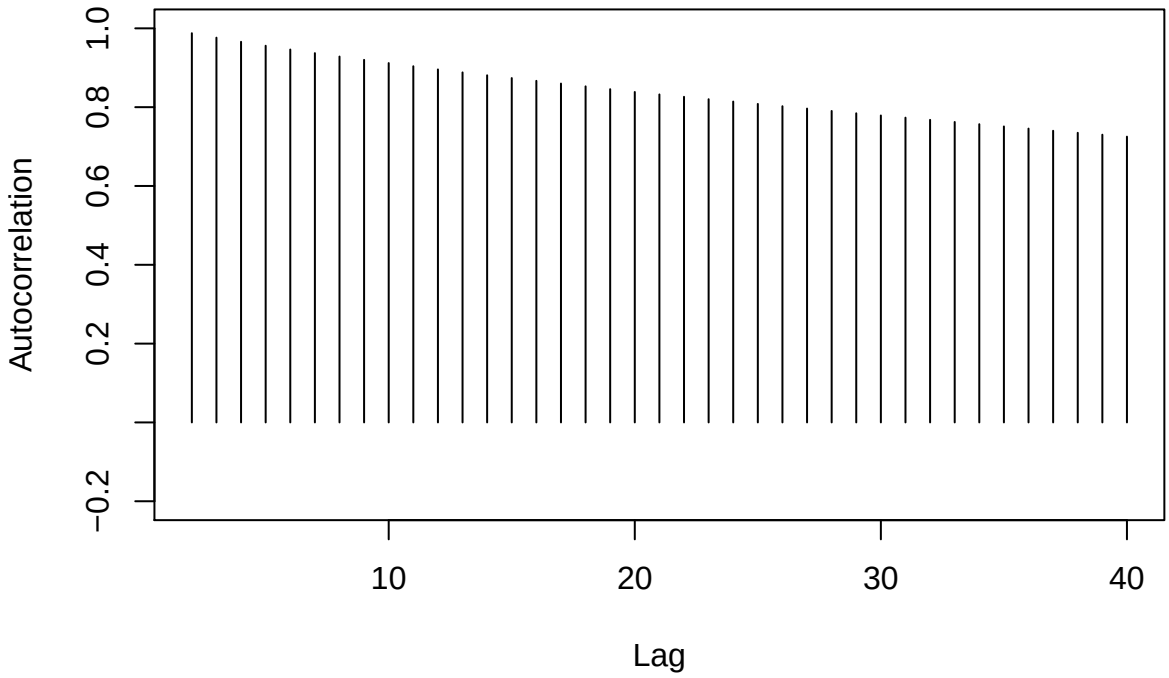
We plot the CUMSUM diagnostic plot next.

```
mu <- mean(mc_data2)
plot(cumsum(mc_data2 - mu), type='l', main='CUMSUM Plot', ylab='')
```



Next, we plot the autocorrelation plot for the chain.

```
n <- 40
plot(seq(2, n, 1), sapply(seq(2, n, 1), function(lag) autocorr(mc_data2, lag)),
     type='h', ylim=c(-.2, 1), xlab="Lag", ylab="Autocorrelation")
```



Based on the three plots above, this chain appears to be somewhat mixing effectively, although not at a rate that would be ideal.

The sample plot shows that the chain is yields points between .2 and .65 with varying frequencies. It seems to have periods of difficulty where it gets stuck in ranges. But it doesn't stay that way for very long.

The CUMSUM plot sticks around zero, especially at first, but eventually deviates far from there. It rebounds back towards zero, but only after a while.

The autocorrelation plot shows decay, but not very quickly.

It seems as though, while the chain is mixing well, it hasn't converged on the stationary distribution.

Problem 5

First, we read in the Markov Chain from the files.

```
chain_list <- lapply(seq(1, 7), function(i) scan(paste0("Module 6 Data Sets/Chain ", i), skip=1))
```

Next, we define the needed expressions. It's all pretty simple algebra straight out of the text.

```
cut_chain <- function(chain, D, L){
  return(chain[D: (D + L)])
}

get_xbar_j <- function(chain_list, D, L){
  return(sapply(chain_list, function(chain){
    mean(cut_chain(chain, D, L))
```

```
  }))
}

get_xbar <- function(chain_list, D, L){
  return(mean(get_xbar_j(chain_list, D, L)))
}

get_B <- function(data_list, D, L){
  J <- length(data_list)
  return((L / (J - 1)) * sum((get_xbar_j(data_list, D, L) - get_xbar(data_list, D, L)) ^ 2))
}

get_s2_j <- function(chain_list, D, L){
  sapply(chain_list, function(chain){
    chain_ <- cut_chain(chain, D, L)
    xbarj <- mean(chain_)
    return((1 / (L - 1)) * sum((chain_ - xbarj)^2))
  })
}

get_W <- function(data_list, D, L){
  mean(get_s2_j(data_list, D, L))
}

get_R <- function(data_list, D, L){
  B <- get_B(data_list, D, L)
  W <- get_W(data_list, D, L)
  return((((L - 1) / L) * W + (1 / L) * B) / W)
}
```

Next, we run the code for each of the values of D and L in the assignment.

```
D <- c(0, 500, 0, 250, 0, 25)
L <- c(1000, 500, 500, 250, 50, 25)
parameters <- data.frame(D=D, L=L)

parameters$sqrt_R <- sapply(seq(1, length(L)), function(i){
  sqrt(get_R(chain_list, D=parameters$D[i], parameters$L[i]))
})

parameters$B <- sapply(seq(1, length(L)), function(i){
  get_B(chain_list, D=parameters$D[i], parameters$L[i])
})

parameters$W <- sapply(seq(1, length(L)), function(i){
  get_W(chain_list, D=parameters$D[i], parameters$L[i])
})

knitr::kable(parameters, col.names = c("D", "L", "sqrt(R)", "B", "W"))
```

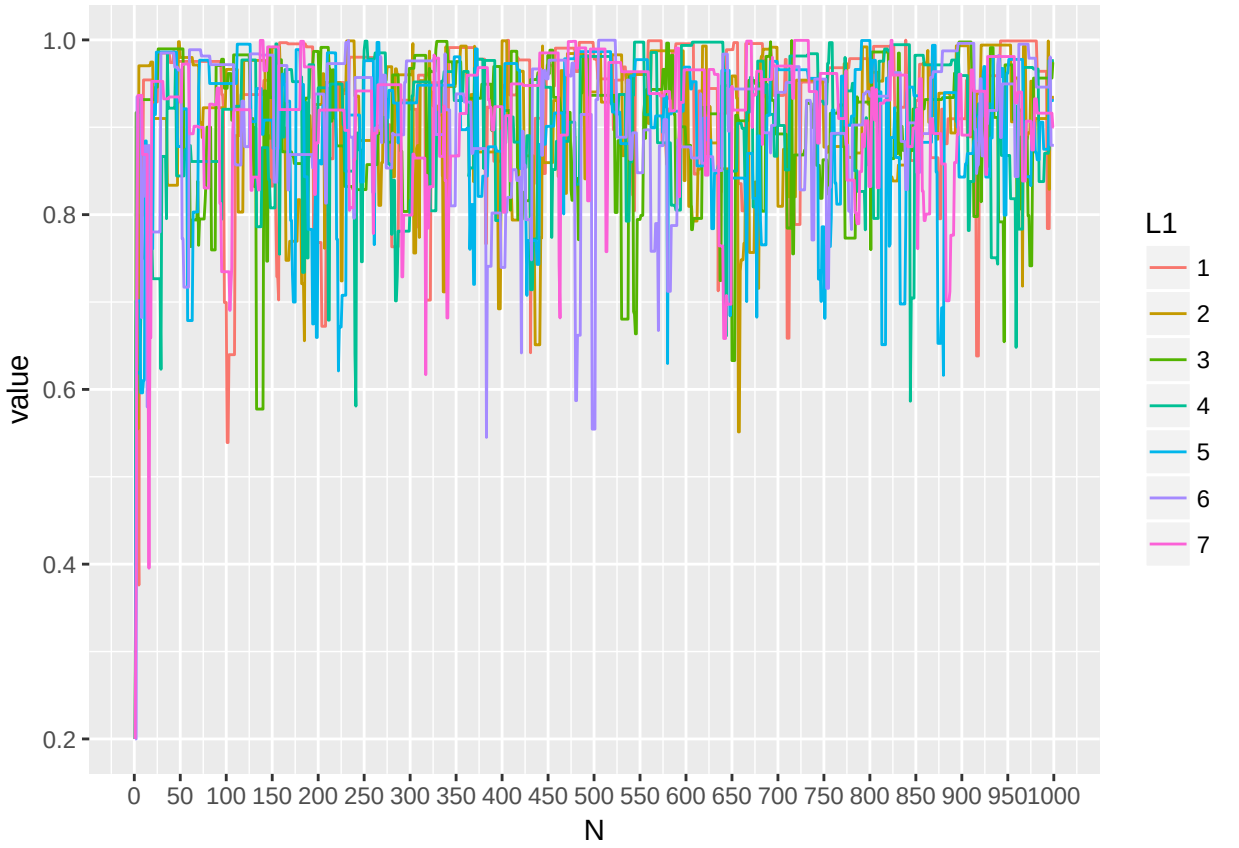
D	L	sqrt(R)	B	W
0	1000	1.008212	0.1137349	0.0065023
500	500	1.021108	0.1086318	0.0048647
0	500	1.006901	0.0636821	0.0080359

D	L	sqrt(R)	B	W
250	250	1.018940	0.0564182	0.0053429
0	50	1.045179	0.1282458	0.0228196
25	25	1.378126	0.0628087	0.0026749

Looking at each of the results, the chain with only 25 observations does not appear to have mixed sufficently by the $\sqrt{R} < 1.1$ standard. However, plotting the paths, we can make some other observations.

```
chain_list <- lapply(chain_list, function(chain){
  data.frame(N=seq(1, length(chain)), value=chain)
})

names(chain_list) <- seq(1, length(chain_list))
data <- melt(chain_list, id.vars=c('N'))
ggplot(data, aes(x=N, y=value, col=L1)) + geom_line() +
  scale_x_continuous(breaks=seq(0, max(data$N), 50))
```



It seems like the results from the first 50 observations are not too great either, as approximately the first 25 observations are heavily influenced by the starting value.

The takeaway here is that even though the $\sqrt{R}$ seems good, it's worth further inspection. The tools laid out in the chapter, graphical and quantitative, can't effectively be used in isolation.