Lab 4

Your Name

MCMC sampling in R

Task 1 (25 points)

In the last lab, I showed that we can create an exact posterior estimate of the mean (θ) of a normally-distributed variate Y, if we have multiple data observations, and if we have a fixed (known) variance. Given that y is a vector of n data observations, then the posterior becomes:

$$\theta|y,\sigma^2 \sim Normal\left(\frac{\frac{\mu_0}{\tau_0^2}+\frac{\bar{y}n}{\sigma^2}}{\frac{1}{\tau_0^2}+\frac{n}{\sigma^2}}, \frac{1}{\frac{1}{\tau_0^2}+\frac{n}{\sigma^2}}\right).$$

Here $\bar{y}$ is the average of our data observations, and again, n is the number of data observations.

Before, we calculated our posterior by creating a sequence of possible θ values and manually calculating the posterior of each value. As I discussed in lecture, this becomes exceedingly inefficient if we have many parameters in our model. Therefore we often use MCMC methods to randomly sample from our unknown posterior distribution.

On the next page, I have provided code for estimating θ from a vector of normally distributed data, with fixed variance, using the MH-MCMC algorithm.

Complete the following:

1. Create in-line comments on the code (using the `#` symbol) to demonstrate your understanding of what is happening in the code (2 points). Be detailed, but not too verbose. Be careful that your comments do not extend the margins of your page. Default to adding newlines rather than creating long lines of text.
2. Run multiple MCMC chains, remove the burn-in ($n/2$) from each chain, and store the resulting output in an R object (e.g., a data frame). Then, create a function to calculate the Gelman-Ruben statistic, based on the equations in class. Use this function to determine if your chains have converged (15 points).
3. Validate that the MCMC sampler correctly approximates the theoretical (i.e., analytically-derived) mean and variance of the posterior distribution of the unknown parameter, θ. Look at the equation above to help (3 points).
4. Conduct some synthetic experiments with your MCMC sampler. Specifically, use your sampler with different prior and proposal distributions. How does the prior and proposal affect the MCMC sampler's efficiency and how quickly it reaches a "good" probability space? Do the same with differently sized data sets. I do not want to see the output of these simulations. Just use the MCMC on your own as a "toy" model. Then, using these anecdotal simulations as support, briefly comment on how the priors, proposals, and data set size influence the MCMC sampler (5 points).

The MH-MCMC sampler for a model with a single parameter, assuming $Y \sim Normal(\theta, \sigma^2)$.

```
#-----------------------------------
#-----------------------------------

mu_true = 3
sd_fixed = 2

mu_prior = 5
sd_prior = 10

mu_prop = 5
sd_prop = sd_prior * 1.1

#-----------------------------------
#-----------------------------------
n_sample = 30
y_test = rnorm(n_sample, mu_true, sd_fixed)

#-----------------------------------
#-----------------------------------

n_iter = 10000
chosen_theta = NULL

for(i in 1:n_iter){

  if(i == 1){
    old_theta = rnorm(1, mu_prop, sd_prop)
  }

  new_theta = rnorm(1, mu_prop, sd_prop)

  old_prop_adj = dnorm(old_theta, mu_prop, sd_prop, log = TRUE)
  new_prop_adj = dnorm(new_theta, mu_prop, sd_prop, log = TRUE)

  old_prior = dnorm(old_theta, mu_prior, sd_prior, log = TRUE)
  new_prior = dnorm(new_theta, mu_prior, sd_prior, log = TRUE)

  old_lik = sum(dnorm(y_test, old_theta, sd_fixed, log = TRUE))
  new_lik = sum(dnorm(y_test, new_theta, sd_fixed, log = TRUE))

  old_post = old_prior + old_lik
  new_post = new_prior + new_lik

  log_ratio = (new_post - new_prop_adj) - (old_post - old_prop_adj)
  ratio = exp(log_ratio)

  if(ratio > 1){
    chosen_theta[i] = new_theta
  }else{

    rand = runif(1, min = 0, max = 1)
```

```
    if(ratio >= rand){
      chosen_theta[i] = new_theta
    }else{
      chosen_theta[i] = old_theta
    }

  }

  old_theta = chosen_theta[i]

}
```

Here are some example plots of the output of a single chain:

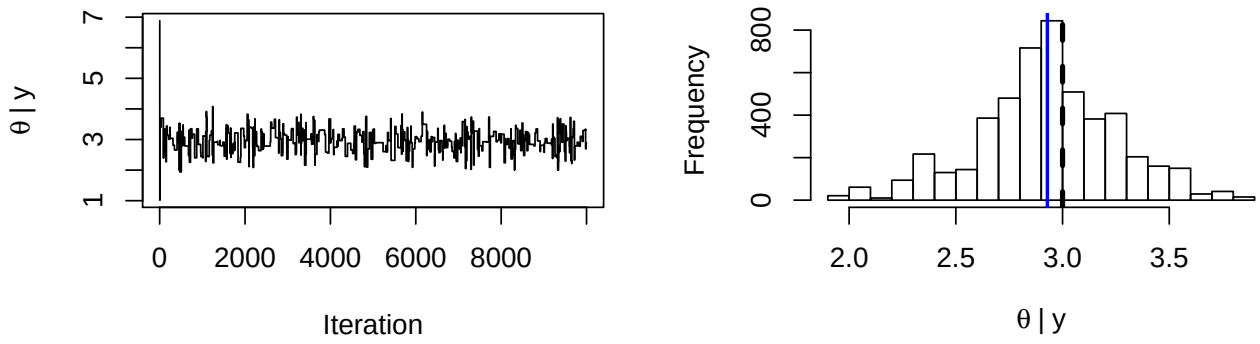
```
par(mfrow = c(1, 2))

# Plot the full trace:
plot(chosen_theta ~ c(1:n_iter), type = "l",
     xlab = "Iteration", ylab = expression(theta~"|"~y))

# Discard the burn-in
# Plot the sample of the estimated posterior of \theta
n_burn = n_iter / 2
hist(chosen_theta[n_burn:n_iter], main = "",
     xlab = expression(theta~"|"~y), ylab = "Frequency")

# Add the mean estimate
mu_est = mean(chosen_theta[n_burn:n_iter])
abline(v=mu_est, col = "blue", lwd = 2)

# Add the true mean
abline(v = mu_true, col = "black", lwd = 3, lty = 2)
```




```
par(mfrow = c(1,1))
```

Task 2 (25 points)

Complete the following:

1. Extend your MCMC sampler to accommodate 2 parameters. Here we will estimate both the mean and the standard deviation of the data (15 points). A hint for the proposal distributions: you can sum the proposal densities of the two parameters together on the log scale to get the total proposal adjustment. Also, be sure to store your output correctly. Your code should store the marginal posteriors of both parameters of interest.

```
#-----------------------------------
#-----------------------------------

mu_true = 3 # Parameter 1 (unknown)
sd_true = 2 # Parameter 2 (unknown)

##### FOR PARAMETER 1
# Prior for Parameter 1
# Normal Distribution:
mu_prior = 5
sd_prior = 10

# Proposal for Parameter 1
# Normal Distribution
mu_prop = 5
sd_prop = sd_prior * 1.1

##### FOR PARAMETER 2
# Prior for Parameter 2
# Gamma distribution:
shape_prior = 1
rate_prior = 0.5

# Proposal for Parameter 2
# Gamma distribution
shape_prop = 1
rate_prop = rate_prior*0.5

#-----------------------------------
#-----------------------------------
n_sample = 30
y_test = rnorm(n_sample, mu_true, sd_true)
#-----------------------------------
#-----------------------------------
```

2. Run multiple MCMC chains, and store the output appropriately in an object. This can be done in a data frame object if you are careful, but `array()`'s are also helpful. You would have $p \times n \times m$ dimensions in the array, where m is the number of chains, n is the number of iterations in each chain, and p is the number of parameters.

3. Discard your burn-in, plot the traces of each parameter, and use your function to calculate the $\hat{R}$ for each parameter (5 points).

4. Calculate the mean, marginal posterior estimates for each parameter. Plot the joint posterior samples of parameter 1 and parameter 2, and place a large "point" at the location of these mean estimates. Hint: You can use the `points()` function. (5 points)