STAT 223-423 Final Capstone

Due Monday, May 2, 2022, at 9 a.m. EDT

Derek Caramella

Please complete this exam using `RMarkdown`. You are not permitted to communicate with anyone other than the instructor regarding the capstone, nor are you permitted to post questions to online forums. You may use any resources from class in answering these problems. Make sure to set a seed to 1234 so all results are reproducible (you should do this once for the entire document).

Type out the academic honesty pledge, and "sign" it by typing your name after it: "I affirm that I will not give or receive any unauthorized help on this exam and that all work will be my own."

Type out pledge here:
I affirm that I will not give or receive any unauthorized help on this exam and that all work will be my own.

Sign by typing name here:
Derek Caramella

An airline company is interested in examining vaccination trends among its travelers. It ultimately wants to know what percentage of its travelers are 1) full vaccinated and boosted, 2) fully vaccinated but not boosted, 3) partially vaccinated, or 4) not vaccinated (i.e. there are $K = 4$ groups). The company wants to examine such vaccination trends both for domestic and international flights (i.e. $J = 2$ groups). They collect data from a random sample of travelers on recent flights and have compiled it in the table below:

Vaccination Status	Fully and Boosted	Fully but not boosted	Partially	Unvaccinated
Domestic flight travelers	12	43	4	13
International flight travelers	23	30	2	11

You've been hired to examine the trends and report back.

To begin your analysis, assume that travelers' vaccination statuses are independent of each other and can be modeled by a multinomial distribution:

$$\mathbf{y_j} \sim Multinomial(\theta_\mathbf{j})$$

Appropriate model priors to assign in this case are:

$$\theta_\mathbf{j}|\alpha \sim Dir(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$$
$$p(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \propto 1$$

As an extension to homework 5, the full appropriate posterior distributions are as follows:

$$\theta_j|\alpha_1, \alpha_2, \alpha_3, \alpha_4, Y \sim Dirichlet(y_{j1} + \alpha_1, y_{j2} + \alpha_2, y_{j3} + \alpha_3, y_{j4} + \alpha_4)$$
$$\alpha_1, \alpha_2, \alpha_3, \alpha_4|Y \propto \prod_{j=1}^{2} \frac{\Gamma(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4)}{\Gamma(\alpha_1)\Gamma(\alpha_2)\Gamma(\alpha_3)\Gamma(\alpha_4)} \frac{\Gamma(y_{j1} + \alpha_1)\Gamma(y_{j2} + \alpha_2)\Gamma(y_{j3} + \alpha_3)\Gamma(y_{j4} + \alpha_4)}{\Gamma(y_{j1} + y_{j2} + y_{j3} + y_{j4} + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4)}$$

1. Draw 10000 samples of each parameter (i.e.$\alpha_1, \alpha_2, \alpha_3, \alpha_4, \theta_\mathbf{1}, \theta_\mathbf{2}$). You can use a proposal variance of `diag(6,4)` and initial values of 1 for each α. **Print the posterior mean for each parameter.** (For reference, my MCMC algorithm is taking about 4.5 minutes to run.)
   a. Grad students only: write your algorithm to allow for a burn-in and for thinning. With this, your function should take two additional arguments, `burnin` and `thin`, that will allow you to specify a burn-in and a thinning option.

```r
y_vector_1 <- c(12, 43, 4, 13)
y_vector_2 <- c(23, 30, 2, 11)
dat1 <- rbind(y_vector_1, y_vector_2)  # Put data into single object
K <- ncol(dat1)  # Number of categories
J <- nrow(dat1)  # Number of groups
initial_alphas <- rep(1, K)  # Initial alphas
n.iter <- 10000
```

Unnormalized Posterior

```r
p <- function(data, alpha_vector) {
    alpha_1 <- alpha_vector[1]
    alpha_2 <- alpha_vector[2]
    alpha_3 <- alpha_vector[3]
    alpha_4 <- alpha_vector[4]
    running_product <- 1
    for (j in 1:J) {
        y_1 <- data[j, 1]
        y_2 <- data[j, 2]
        y_3 <- data[j, 3]
        y_4 <- data[j, 4]
        instance_comp <- ((gamma(alpha_1 + alpha_2 + alpha_3 +
            alpha_4))/(gamma(alpha_1) * gamma(alpha_2) * gamma(alpha_3) *
            gamma(alpha_4))) * (((gamma(alpha_1 + y_1)) * (gamma(alpha_2 +
            y_2)) * (gamma(alpha_3 + y_3)) * (gamma(alpha_4 +
            y_4)))/(gamma(y_1 + y_2 + y_3 + y_4 + alpha_1 + alpha_2 +
            alpha_3 + alpha_4)))

        running_product <- running_product * instance_comp
    }
    return(running_product)
}
```

Proposal Distribution

```r
Q <- function(alpha_mean_vector, prop.var) {
    rtmvnorm(1, mu = alpha_mean_vector, sigma = prop.var, lb = c(0,
        0, 0, 0))
}
```

```r
qdens <- function(alpha_vector, alpha_mean_vector, prop.var) {
    dtmvnorm(alpha_vector, mu = alpha_mean_vector, sigma = prop.var,
        lb = c(0, 0, 0, 0))  #sigma is the variance, not the SD
}
```

4-Dimensional Metropolis-Hastings Algorithm

```r
mh4D <- function(init_alphas, data, p, Q, qdens, prop.var, n.iter,
    thin = 1, burn_in = 0) {
    results <- matrix(nrow = n.iter, ncol = 16)
    colnames(results) <- c("t", "alpha_1.t", "alpha_1.new", "alpha_2.t",
        "alpha_2.new", "alpha_3.t", "alpha_3.new", "alpha_4.t",
        "alpha_4.new", "alpha", "u", "accept", "decision_alpha_1",
        "decision_alpha_2", "decision_alpha_3", "decision_alpha_4")

    # Initialize Chain
    alpha_vector.t <- init_alphas

    for (t in 1:n.iter) {
        # Draw proposed value of (alpha_1, alpha_2,
        # alpha_3, alpha_4)
        new <- Q(alpha_vector.t, prop.var)

        # calculate acceptance probability
        u <- runif(1, 0, 1)
        alpha <- min(1, (p(data, new)/p(data, alpha_vector.t)) *
            (qdens(alpha_vector.t, new, prop.var)/qdens(new,
                alpha_vector.t, prop.var)))

        # Decide whether to accept or reject
        if (u <= alpha) {
            value <- new  # Accept the new alphas
        } else {
            value <- alpha_vector.t  # Stay at current values of alphas
        }

        # store all your information
        results[t, "t"] <- t
        results[t, "alpha_1.t"] <- alpha_vector.t[1]
        results[t, "alpha_1.new"] <- new[1]
        results[t, "alpha_2.t"] <- alpha_vector.t[2]
        results[t, "alpha_2.new"] <- new[2]
        results[t, "alpha_3.t"] <- alpha_vector.t[3]
        results[t, "alpha_3.new"] <- new[3]
        results[t, "alpha_4.t"] <- alpha_vector.t[4]
        results[t, "alpha_4.new"] <- new[4]
        results[t, "alpha"] <- alpha
        results[t, "u"] <- u
        results[t, "accept"] <- ifelse(u <= alpha, T, F)
        results[t, "decision_alpha_1"] <- value[1]
        results[t, "decision_alpha_2"] <- value[2]
        results[t, "decision_alpha_3"] <- value[3]
        results[t, "decision_alpha_4"] <- value[4]

        # Update values for next iteration
        alpha_vector.t <- value
    }

    # Burn In
    if (burn_in != 0) {
        results <- tail(results, -burn_in)
    }

    # Thining
    results <- cbind(results, 1:nrow(results))
    results <- results[(results[, 17]%%thin) == 0, ]
    results <- results[, 1:16]

    return(results)
}
```

Run Algorithm

```
alpha_draws_complete_report <- mh4D(init_alphas = initial_alphas,
    data = dat1, p = p, Q = Q, qdens = qdens, prop.var = diag(6,
        4), n.iter = n.iter)
alpha_draws <- alpha_draws_complete_report[, 13:16]

head(alpha_draws, 3)
```

```
##      decision_alpha_1 decision_alpha_2 decision_alpha_3 decision_alpha_4
## [1,]          1.67956         3.656328         2.051137         2.239579
## [2,]          1.67956         3.656328         2.051137         2.239579
## [3,]          1.67956         3.656328         2.051137         2.239579
```

Sample Thetas

```
posterior_theta_vector_1 <- matrix(NA, n.iter, K)
posterior_theta_vector_2 <- matrix(NA, n.iter, K)

for (n in 1:n.iter) {
    posterior_theta_vector_1[n, ] <- rdirichlet(1, c(y_vector_1[1] +
        alpha_draws[n, "decision_alpha_1"], y_vector_1[2] + alpha_draws[n,
        "decision_alpha_2"], y_vector_1[3] + alpha_draws[n, "decision_alpha_3"],
        y_vector_1[4] + alpha_draws[n, "decision_alpha_4"]))
    posterior_theta_vector_2[n, ] <- rdirichlet(1, c(y_vector_2[1] +
        alpha_draws[n, "decision_alpha_1"], y_vector_2[2] + alpha_draws[n,
        "decision_alpha_2"], y_vector_2[3] + alpha_draws[n, "decision_alpha_3"],
        y_vector_2[4] + alpha_draws[n, "decision_alpha_4"]))
}
```

```
posterior_alphas_report <- t(apply(alpha_draws, 2, mean))
colnames(posterior_alphas_report) <- c("Alpha 1", "Alpha 2",
    "Alpha 3", "Alpha 4")
cat("Posterior Mean Alpha Report")
cat("\n")
kable(posterior_alphas_report, "markdown")
cat("\n")

posterior_thetas_report <- t(apply(posterior_theta_vector_1,
    2, mean))
colnames(posterior_thetas_report) <- c("Theta1,1", "Theta1,2",
    "Theta1,3", "Theta1,4")

posterior_thetas_report_2 <- t(apply(posterior_theta_vector_2,
    2, mean))
colnames(posterior_thetas_report_2) <- c("Theta2,1", "Theta2,2",
    "Theta2,3", "Theta2,4")

posterior_thetas_report <- round(cbind(posterior_thetas_report,
    posterior_thetas_report_2), 3)

cat("Posterior Mean Theta Report")
cat("\n")
kable(posterior_thetas_report, "markdown")
```

```
## Posterior Mean Alpha Report
```

Alpha 1	Alpha 2	Alpha 3	Alpha 4
19.58239	40.01386	4.442395	13.77101

```
##
## Posterior Mean Theta Report
```

Theta1,1	Theta1,2	Theta1,3	Theta1,4	Theta2,1	Theta2,2	Theta2,3	Theta2,4
0.21	0.555	0.057	0.179	0.297	0.486	0.045	0.173

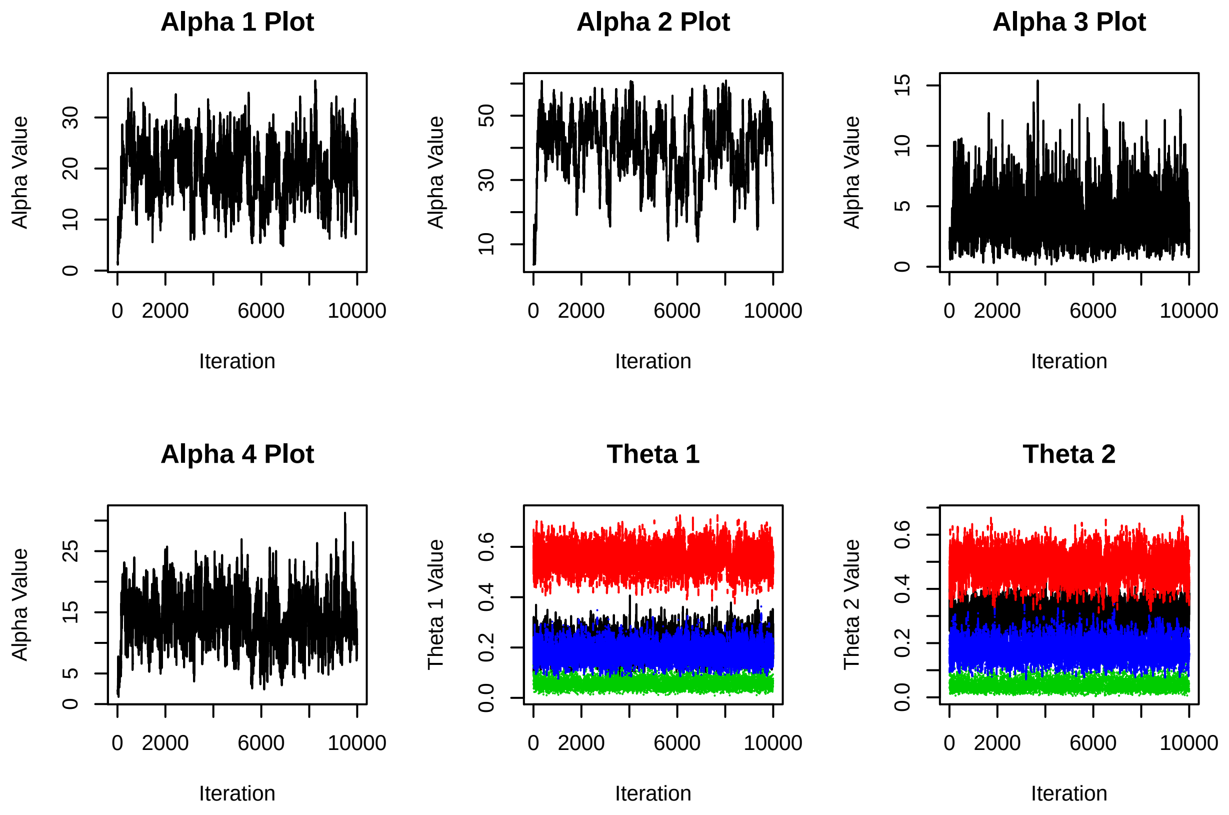
2. Construct and examine traceplots for each of the parameters, and discuss whether you believe convergence has been met. For each θ_j parameter, you should put all $K = 4$ values on one traceplot in different colors (i.e. in total, you will have one traceplot for each alpha, one traceplot for θ_1 with four traces on it, and one traceplot for θ_2 with four traces on it). Plot the traceplots on a 2×3 grid of plots (i.e. `par(mfrow=c(2,3))`).

```r
par(mfrow = c(2, 3))


# Plot Alphas
for (i in 1:4) {
    plot(alpha_draws[, i], type = "l", main = paste("Alpha",
        i, "Plot", sep = " "), ylab = "Alpha Value", xlab = "Iteration")
}

# Plots Theta 1
matplot(posterior_theta_vector_1, type = "l", main = "Theta 1",
    xlab = "Iteration", ylab = "Theta 1 Value", col = 1:4)

# Plots Theta 2
matplot(posterior_theta_vector_2, type = "l", main = "Theta 2",
    xlab = "Iteration", ylab = "Theta 2 Value", col = 1:4)
```



When analyzing trace plots, we are looking for “bushy hedges” to conclude convergence. Alpha 1 and Alpha 2 are weak bushy hedges; however, Alpha 3 and Alpha 4 have strong bushy hedges. Therefore, we need more information to conclude convergence for Alpha 1 and Alpha 2, we will look at the Raftery-Lewis diagnostic in a later section. The theta variables appear to converge since each variable produced a bushy hedge.

3. Calculate the effective sample size for each parameter.

```r
cat("Alphas")
cat("\n")
effectiveSize(alpha_draws)
cat("\n")
cat("Theta 1")
cat("\n")
effectiveSize(posterior_theta_vector_1)
cat("\n")
cat("Theta 2")
cat("\n")
effectiveSize(posterior_theta_vector_2)
```

```
## Alphas
## decision_alpha_1 decision_alpha_2 decision_alpha_3 decision_alpha_4
##         202.4174          78.4959         638.7888         264.4779
##
## Theta 1
##      var1      var2      var3      var4
##  758.6443 1055.3034 2721.8854 1316.4621
##
## Theta 2
##      var1      var2      var3      var4
##  746.8553  716.6997 2377.0322 1260.4003
```

The effective sample sizes are relatively large except for Alpha 2 and Alpha 1. Further analysis is required to conclude convergence.

4. Calculate the acceptance rate for Metropolis-Hastings algorithm. Briefly comment on this value.

```r
mean(alpha_draws_complete_report[, "accept"])
```

```
## [1] 0.5266
```

The current Metropolis-Hastings algorithm is multidimensional, hence we are targeting acceptance rate of roughly 25%. In our sampled chain, we observed an 52.66% acceptance rate, which is less than 25%. Therefore, we are staying too often in one area and not exploring too much. One approach to increasing the acceptance rate is by increasing proposed variance.

5. Calculate the Raftery-Lewis diagnostic for each parameter, using $r = 0.0125$. Comment on the results.

```r
cat("Alphas")
cat("\n")
raftery.diag(alpha_draws, r = 0.0125)
```

```r
cat("\n")
cat("Theta 1")
cat("\n")
raftery.diag(posterior_theta_vector_1, r = 0.0125)
cat("\n")
cat("Theta 2")
cat("\n")
raftery.diag(posterior_theta_vector_2, r = 0.0125)
```

```
## Alphas
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.0125
## Probability (s) = 0.95
##
##                   Burn-in  Total Lower bound  Dependence
##                   (M)      (N)   (Nmin)       factor (I)
##  decision_alpha_1 41       7071  600          11.80
##  decision_alpha_2 192      27952 600          46.60
##  decision_alpha_3 10       1749  600           2.92
##  decision_alpha_4 72       12056 600          20.10
##
##
## Theta 1
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.0125
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  8        1490  600          2.48
##  6        1456  600          2.43
##  3        674   600          1.12
##  6        1552  600          2.59
##
##
## Theta 2
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.0125
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)      factor (I)
##  6        1450  600         2.42
##  8        1494  600         2.49
##  3        680   600         1.13
##  6        1302  600         2.17
```

The chain length is 10,000 – when evaluating the Raftery-Lewis diagnostic, we are looking for the total N to be less than the chain length (10,000) to conclude convergence. The only variables that did not converge were Alpha 2 (27,952) and Alpha 4 (12,056). Alpha 4 is close to convergence by missing roughly 2,000 samples. However, Alpha 2 requires nearly double the number of samples. Although Alpha 2 and Alpha 4 did not converge, the remaining variables did converge; therefore, we can conclude that we have sufficient samples; however, timeline and resource permitting, we should collect more samples.

6. Report the posterior mean and 95% posterior interval for each variable. Present the results in a nicely formatted table, rounding to three decimal places.

```r
posterior_alphas_report <- round(rbind(apply(alpha_draws, 2,
    mean), apply(alpha_draws, 2, quantile, c(0.025, 0.975))),
    3)
rownames(posterior_alphas_report) <- c("Mean", "2.5%", "97.5%")
colnames(posterior_alphas_report) <- c("Alpha 1", "Alpha 2",
    "Alpha 3", "Alpha 4")
cat("Posterior Alpha Report")
cat("\n")
kable(posterior_alphas_report, "markdown")
cat("\n")

posterior_thetas_report <- rbind(apply(posterior_theta_vector_1,
    2, mean), apply(posterior_theta_vector_1, 2, quantile, c(0.025,
    0.975)))
rownames(posterior_thetas_report) <- NULL
colnames(posterior_thetas_report) <- c("Theta1,1", "Theta1,2",
    "Theta1,3", "Theta1,4")

posterior_thetas_report_2 <- rbind(apply(posterior_theta_vector_2,
    2, mean), apply(posterior_theta_vector_2, 2, quantile, c(0.025,
    0.975)))
rownames(posterior_thetas_report_2) <- NULL
colnames(posterior_thetas_report_2) <- c("Theta2,1", "Theta2,2",
    "Theta2,3", "Theta2,4")

posterior_thetas_report <- round(cbind(posterior_thetas_report,
    posterior_thetas_report_2), 3)
rownames(posterior_thetas_report) <- c("Mean", "2.5%", "97.5%")
```

```r
cat("Posterior Theta Report")
cat("\n")
kable(posterior_thetas_report, "markdown")
```

```
## Posterior Alpha Report
```

	Alpha 1	Alpha 2	Alpha 3	Alpha 4
Mean	19.582	40.014	4.442	13.771
2.5%	7.844	17.392	1.144	5.201
97.5%	30.268	56.020	9.591	22.225

```
##
## Posterior Theta Report
```

	Theta1,1	Theta1,2	Theta1,3	Theta1,4	Theta2,1	Theta2,2	Theta2,3	Theta2,4
Mean	0.210	0.555	0.057	0.179	0.297	0.486	0.045	0.173
2.5%	0.129	0.458	0.020	0.112	0.210	0.386	0.012	0.104
97.5%	0.298	0.650	0.109	0.256	0.393	0.586	0.097	0.253

7. Interpret the posterior interval for θ_{12}. Be careful to think exactly what this is describing by itself.

For a Domestic Flight, the probability of falling into the Fully but not boosted category is 55.5 percent, on average. The 95% confidence interval suggests the probability of falling into the Fully but not boosted category ranges from 0.458 percent to 0.650 percent.

8. Take 10000 posterior draws of $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$ (i.e. the posterior predictive distribution. Note, though, that you do not need to derive anything…think about how we took draws from the posterior predictive distribution in Assignment 4).

```r
# Our n comes from question 9
y_1_tilde <- rmultinomial(n = 170, posterior_theta_vector_1)
y_2_tilde <- rmultinomial(n = 200, posterior_theta_vector_2)

colnames(y_1_tilde) <- c("Fully and Boosted", "Fully but not boosted",
    "Partially", "Unvaccinated")
colnames(y_2_tilde) <- c("Fully and Boosted", "Fully but not boosted",
    "Partially", "Unvaccinated")
```

9. Given that domestic flights are of size 170 passengers and international flights are of size 200 passengers, calculate the posterior probability (separately for domestic and international flights) that a future flight will have more than 20% of its passengers being unvaccinated.

```
cat(paste("The posterior probability for domestic flights that a future flight will have
    round(mean((y_1_tilde[, "Unvaccinated"]/apply(y_1_tilde,
        1, sum)) > 0.2) * 100, 3)[1], "%.", sep = ""))

cat("\n")

cat(paste("The posterior probability for international flights that a future flight will
    round(mean((y_2_tilde[, "Unvaccinated"]/apply(y_2_tilde,
        1, sum)) > 0.2) * 100, 3)[1], "%.", sep = ""))
```

```
## The posterior probability for domestic flights that a future flight will have more th
## The posterior probability for international flights that a future flight will have mo
```

The posterior probability for domestic flights that a future flight will have more than 20% of its passengers being unvaccinated is 29.91%.

The posterior probability for international flights that a future flight will have more than 20% of its passengers being unvaccinated is 25.44%.

10. Nicely summarize in terms that the airline company executives (who has a very limited statistics background) will understand what you did, what results you arrived at, and what conclusions you have for them. This write-up should be a short paragraph or two and provide a robust discussion of what you did in laypersons terms.

Two groups exist, (1) Domestic Flight Travelers & (2) International Flight Travelers; within the two groups, four categories exist, (1) Fully and Boosted, (2) Fully but not boosted, (3) Partially, and (4) Unvaccinated. You collected a random sample of travelers for each group and category. We would like to share information between groups one and two becuase they are similar; therefore, we will take draws to combine information from both groups to draw inference. After we collected the shared information, we passed the data through a statistics distribution that returns probabilities for each group and category. We ensured that the results are valid by using multiple statistical tests to ensure we have enough samples. These results are convient becuase we can forecast the number of passengers that belong to a certain group and category.

11. Grad students only: Rerun your algorithm in question 1, allowing for a burn-in of 100 and saving only every 10th value (i.e. `thin=10`). Repeat questions 2-6 for this chain. Comment on the differences in results compared to your original analysis.

Question 1

```
alpha_draws_complete_report <- mh4D(init_alphas = initial_alphas,
    data = dat1, p = p, Q = Q, qdens = qdens, prop.var = diag(6,
        4), n.iter = n.iter, thin = 10, burn_in = 100)
alpha_draws <- alpha_draws_complete_report[, 13:16]

head(alpha_draws, 3)
```

```
##          decision_alpha_1 decision_alpha_2 decision_alpha_3
##   [110,]         14.79908         47.27076         3.180096
##   [120,]         18.85036         36.52111         6.086638
##   [130,]         25.06929         39.76496         3.169012
##          decision_alpha_4
##   [110,]         15.64365
##   [120,]         14.40219
##   [130,]         14.71892
```

```
posterior_theta_vector_1 <- matrix(NA, nrow(alpha_draws), K)
posterior_theta_vector_2 <- matrix(NA, nrow(alpha_draws), K)

for (n in 1:nrow(alpha_draws)) {
    posterior_theta_vector_1[n, ] <- rdirichlet(1, c(y_vector_1[1] +
        alpha_draws[n, "decision_alpha_1"], y_vector_1[2] + alpha_draws[n,
        "decision_alpha_2"], y_vector_1[3] + alpha_draws[n, "decision_alpha_3"],
        y_vector_1[4] + alpha_draws[n, "decision_alpha_4"]))
    posterior_theta_vector_2[n, ] <- rdirichlet(1, c(y_vector_2[1] +
        alpha_draws[n, "decision_alpha_1"], y_vector_2[2] + alpha_draws[n,
        "decision_alpha_2"], y_vector_2[3] + alpha_draws[n, "decision_alpha_3"],
        y_vector_2[4] + alpha_draws[n, "decision_alpha_4"]))
}
```

```
posterior_alphas_report <- t(apply(alpha_draws, 2, mean))
colnames(posterior_alphas_report) <- c("Alpha 1", "Alpha 2",
    "Alpha 3", "Alpha 4")
cat("Posterior Mean Alpha Report")
cat("\n")
kable(posterior_alphas_report, "markdown")
cat("\n")

posterior_thetas_report <- t(apply(posterior_theta_vector_1,
    2, mean))
colnames(posterior_thetas_report) <- c("Theta1,1", "Theta1,2",
    "Theta1,3", "Theta1,4")

posterior_thetas_report_2 <- t(apply(posterior_theta_vector_2,
    2, mean))
colnames(posterior_thetas_report_2) <- c("Theta2,1", "Theta2,2",
    "Theta2,3", "Theta2,4")

posterior_thetas_report <- round(cbind(posterior_thetas_report,
    posterior_thetas_report_2), 3)

cat("Posterior Mean Theta Report")
```

```r
cat("\n")
kable(posterior_thetas_report, "markdown")
```

```
## Posterior Mean Alpha Report
```

Alpha 1	Alpha 2	Alpha 3	Alpha 4
19.84543	41.37269	4.620682	14.66449

```
##
## Posterior Mean Theta Report
```

Theta1,1	Theta1,2	Theta1,3	Theta1,4	Theta2,1	Theta2,2	Theta2,3	Theta2,4
0.208	0.555	0.057	0.181	0.293	0.486	0.046	0.176

For each sampled alpha, the mean is roughly congruent to the vanilla metropolis-hastings algorithm, only differencing by one or two integers. Similarly, posterior thetas are roughly equal to the vanilla approach.

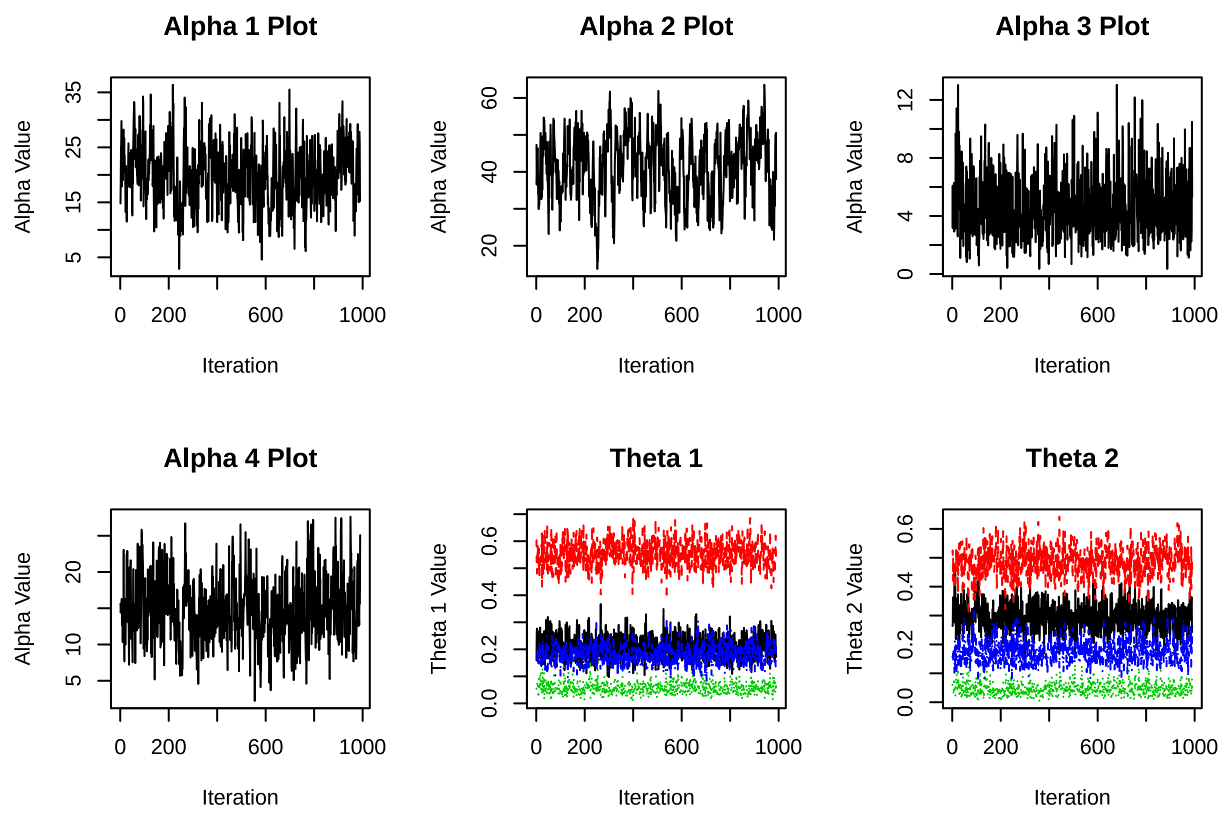
Question 2

```r
par(mfrow = c(2, 3))


# Plot Alphas
for (i in 1:4) {
    plot(alpha_draws[, i], type = "l", main = paste("Alpha",
        i, "Plot", sep = " "), ylab = "Alpha Value", xlab = "Iteration")
}

# Plots Theta 1
matplot(posterior_theta_vector_1, type = "l", main = "Theta 1",
    xlab = "Iteration", ylab = "Theta 1 Value", col = 1:4)

# Plots Theta 2
matplot(posterior_theta_vector_2, type = "l", main = "Theta 2",
    xlab = "Iteration", ylab = "Theta 2 Value", col = 1:4)
```



Alpha 1, Alpha 2, and Alpha 4 do not appear to have a bushy hedge, hence I do not believe these variables converged. Relative to the vanilla approach, Alpha 2 and Alpha 4 did not converge. Similar to both approaches, all variables within Theta 1 and Theta 2 appear to converge due to the bushy hedge.

Question 3

```r
cat("Alphas")
cat("\n")
effectiveSize(alpha_draws)
cat("\n")
cat("Theta 1")
cat("\n")
effectiveSize(posterior_theta_vector_1)
cat("\n")
cat("Theta 2")
cat("\n")
effectiveSize(posterior_theta_vector_2)
```

```
## Alphas
## decision_alpha_1 decision_alpha_2 decision_alpha_3 decision_alpha_4
##        230.56593         95.58764        696.08054        262.99686
```

```
##
## Theta 1
##      var1     var2     var3     var4
## 637.7236 578.2547 755.8750 670.0113
##
## Theta 2
##      var1     var2     var3     var4
## 482.4426 399.6333 769.6438 691.0225
```

Across all variables, the effective sample size was relatively large, I believe this is a result of thining because samples are less dependent on previous record. In both approaches, the effective sample size was relatively large.

Question 4

```
mean(alpha_draws_complete_report[, "accept"])
```

```
## [1] 0.5484848
```

The current Metropolis-Hastings algorithm is multidimensional, hence we are targeting acceptance rate of roughly 25%. In our sampled chain, we observed an 54.84% acceptance rate, which is less than 25% but greater than the vanilla 52.66%. Therefore, we are staying too often in one area and not exploring too much. One approach to increasing the acceptance rate is by increasing proposed variance.

Question 5

```
cat("Alphas")
cat("\n")
raftery.diag(alpha_draws, r = 0.0125)
cat("\n")
cat("Theta 1")
cat("\n")
raftery.diag(posterior_theta_vector_1, r = 0.0125)
cat("\n")
cat("Theta 2")
cat("\n")
raftery.diag(posterior_theta_vector_2, r = 0.0125)
```

```
## Alphas
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.0125
## Probability (s) = 0.95
##
##                      Burn-in  Total Lower bound  Dependence
##                      (M)      (N)   (Nmin)       factor (I)
##  decision_alpha_1 5        954   600          1.590
##  decision_alpha_2 9        1512  600          2.520
##  decision_alpha_3 2        578   600          0.963
##  decision_alpha_4 3        740   600          1.230
##
##
## Theta 1
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.0125
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  3        740   600          1.230
##  2        578   600          0.963
##  2        627   600          1.040
##  2        627   600          1.040
##
##
## Theta 2
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.0125
## Probability (s) = 0.95
##
##  Burn-in  Total Lower bound  Dependence
##  (M)      (N)   (Nmin)       factor (I)
##  3        740   600          1.230
##  4        805   600          1.340
##  2        578   600          0.963
##  2        627   600          1.040
```

The chain length is 990 – when evaluating the Raftery-Lewis diagnostic, we are looking for the total N to be less than the chain length (990) to conclude convergence. The only variables that did not converge were Alpha 1 (1,247), Alpha 2 (1,866), and Alpha 4 (1,137). Relative to the vanilla approach, where Alpha 2 and Alpha 4 did not converge. Due to thinning, we would need to increase our sampling by nearly double to converge on Alpha 1, Alpha 2, and Alpha 4. However, all our theta variables are converged, so it is a resource question to extend the chain, I would recommend on extending the chain.

Question 6

```
posterior_alphas_report <- round(rbind(apply(alpha_draws, 2,
    mean), apply(alpha_draws, 2, quantile, c(0.025, 0.975))),
```

```r
    3)
rownames(posterior_alphas_report) <- c("Mean", "2.5%", "97.5%")
colnames(posterior_alphas_report) <- c("Alpha 1", "Alpha 2",
    "Alpha 3", "Alpha 4")
cat("Posterior Alpha Report")
cat("\n")
kable(posterior_alphas_report, "markdown")
cat("\n")

posterior_thetas_report <- rbind(apply(posterior_theta_vector_1,
    2, mean), apply(posterior_theta_vector_1, 2, quantile, c(0.025,
    0.975)))
rownames(posterior_thetas_report) <- NULL
colnames(posterior_thetas_report) <- c("Theta1,1", "Theta1,2",
    "Theta1,3", "Theta1,4")

posterior_thetas_report_2 <- rbind(apply(posterior_theta_vector_1,
    2, mean), apply(posterior_theta_vector_1, 2, quantile, c(0.025,
    0.975)))
rownames(posterior_thetas_report_2) <- NULL
colnames(posterior_thetas_report_2) <- c("Theta2,1", "Theta2,2",
    "Theta2,3", "Theta2,4")

posterior_thetas_report <- round(cbind(posterior_thetas_report,
    posterior_thetas_report_2), 3)
rownames(posterior_thetas_report) <- c("Mean", "2.5%", "97.5%")

cat("Posterior Theta Report")
cat("\n")
kable(posterior_thetas_report, "markdown")
```

```
## Posterior Alpha Report
```

	Alpha 1	Alpha 2	Alpha 3	Alpha 4
Mean	19.845	41.373	4.621	14.664
2.5%	9.549	24.558	1.364	6.969
97.5%	30.483	56.304	9.433	24.114

```
##
## Posterior Theta Report
```

	Theta1,1	Theta1,2	Theta1,3	Theta1,4	Theta2,1	Theta2,2	Theta2,3	Theta2,4
Mean	0.208	0.555	0.057	0.181	0.208	0.555	0.057	0.181

	Theta1,1	Theta1,2	Theta1,3	Theta1,4	Theta2,1	Theta2,2	Theta2,3	Theta2,4
2.5%	0.133	0.457	0.020	0.113	0.133	0.457	0.020	0.113
97.5%	0.292	0.650	0.106	0.258	0.292	0.650	0.106	0.258

The burn-in and thinning approach yielded slightly different variables means and confidence interval values. However, the alpha values difference by no more than 1.5. Additionally, we see differing theta variables, when evaluating the theta differences, we need to be more stringent because a small change in theta can have larger effects. The vanilla approach differs by roughly a tenths place on each variable. I recommend running the burn-in and thinning approach on more iterations to better guage the true distribution. However, the current results should be valid per our Raftery diagnostic analysis.