ETC3430 Assignment 1

Chelaka Paranahewa

Question 1

Part A

```r
P <- matrix(
    data = c(0.45, 0.35, 0.20, 0.36, 0.34, 0.30, 0.25, 0.65, 0.10),
    nrow = 3, ncol = 3, byrow = TRUE
)
dimnames(P) <- list(
    c("Sunny", "Rainy", "Overcast"),
    c("Sunny", "Rainy", "Overcast")
)
```

Estimated Transition Matrix

Table 1: Estimated Transition Matrix

	Sunny	Rainy	Overcast
Sunny	0.45	0.35	0.2
Rainy	0.36	0.34	0.3
Overcast	0.25	0.65	0.1

Part B

```r
table_out <- tibble()
mode_out <- tibble()

for (day in 0:5) {
    tm <- t(P %^% day)
    table_out <- table_out %>% rbind(tibble(
        "Day"       = day,
        "Sunny"     = glue("[{tm[1, 1]}  {tm[1, 2]}   {tm[1, 3]}]"),
        "Rainy"     = glue("[{tm[2, 1]}  {tm[2, 2]}   {tm[2, 3]}]"),
        "Overcast"  = glue("[{tm[3, 1]}  {tm[3, 2]}   {tm[3, 3]}]")
    ))
    mode_out <- mode_out %>% rbind(tibble(
        "Day" = day,
        "Sunny" = switch(which.max(tm[1, ]),
            "Sunny",
            "Rainy",
            "Overcast"
        ),
```

```
        "Rainy" = switch(which.max(tm[2, ]),
            "Sunny",
            "Rainy",
            "Overcast"
        ),
        "Overcast" = switch(which.max(tm[3, ]),
            "Sunny",
            "Rainy",
            "Overcast"
        )
    ))
}
```

Day	Sunny	Rainy	Overcast
0	[1 0 0]	[0 1 0]	[0 0 1]
1	[0.45 0.36 0.25]	[0.35 0.34 0.65]	[0.2 0.3 0.1]
2	[0.3785 0.3594 0.3715]	[0.4065 0.4366 0.3735]	[0.215 0.204 0.255]
3	[0.370415 0.369906 0.365385]	[0.410435 0.406834 0.422765]	[0.21915 0.22326 0.21185]
4	[0.36923085 0.36873294 0.36958115]	[0.41164065 0.41290966 0.40932735]	[0.2191285 0.2183574 0.2210915]
5	[0.3691266415 0.3691666506 0.3689422385]	[0.4116221435 0.4113781234 0.4122341765]	[0.219251215 0.219455226 0.218823585]

Day	Sunny	Rainy	Overcast
0	Sunny	Rainy	Overcast
1	Sunny	Overcast	Rainy
2	Sunny	Rainy	Overcast
3	Sunny	Overcast	Rainy
4	Overcast	Rainy	Overcast
5	Rainy	Overcast	Rainy

Part C

```
table_out <- tibble()
for (day in 0:5) {
    tm <- t((P %^% day) %*% matrix(1:3))
    table_out <- table_out %>% rbind(tibble(
        "Day" = day,
        "Sunny" = tm[1, 1],
        "Rainy" = tm[1, 2],
        "Overcast" = tm[1, 3]
    ))
}
```

Day	Sunny	Rainy	Overcast
0	1.000000	2.000000	3.000000
1	1.750000	1.940000	1.850000
2	1.836500	1.844600	1.883500
3	1.848735	1.853354	1.846465

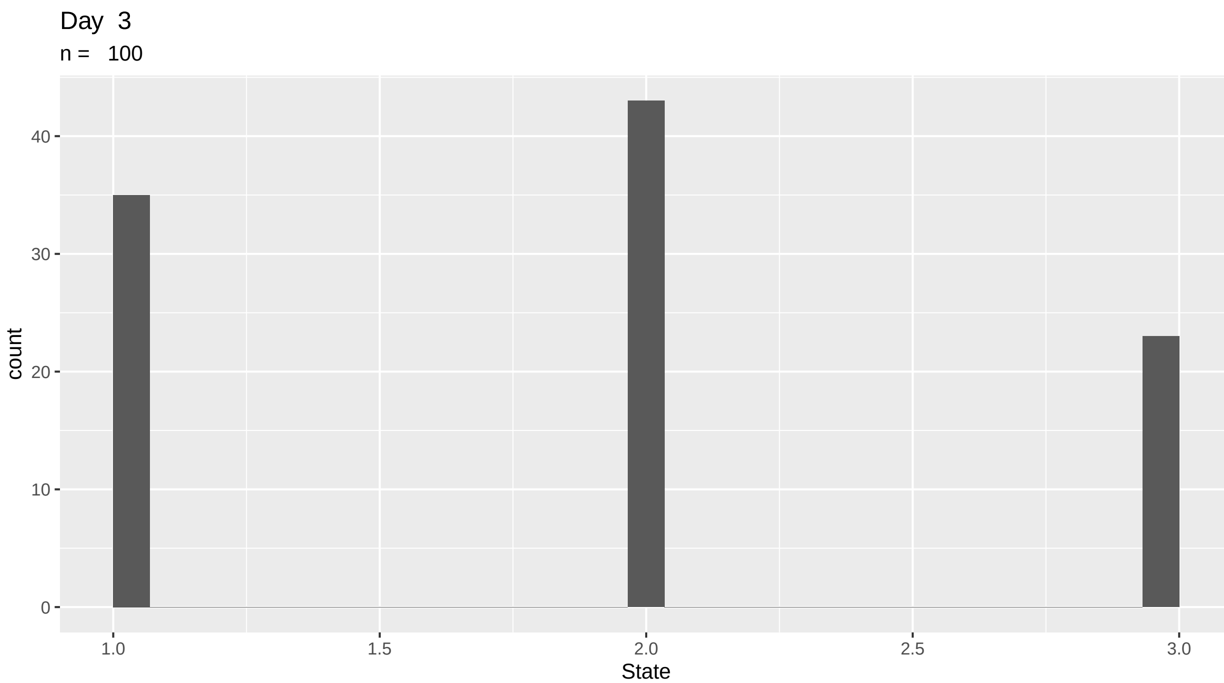
Day	Sunny	Rainy	Overcast
4	1.849898	1.849625	1.851510
5	1.850125	1.850289	1.849881

Part D

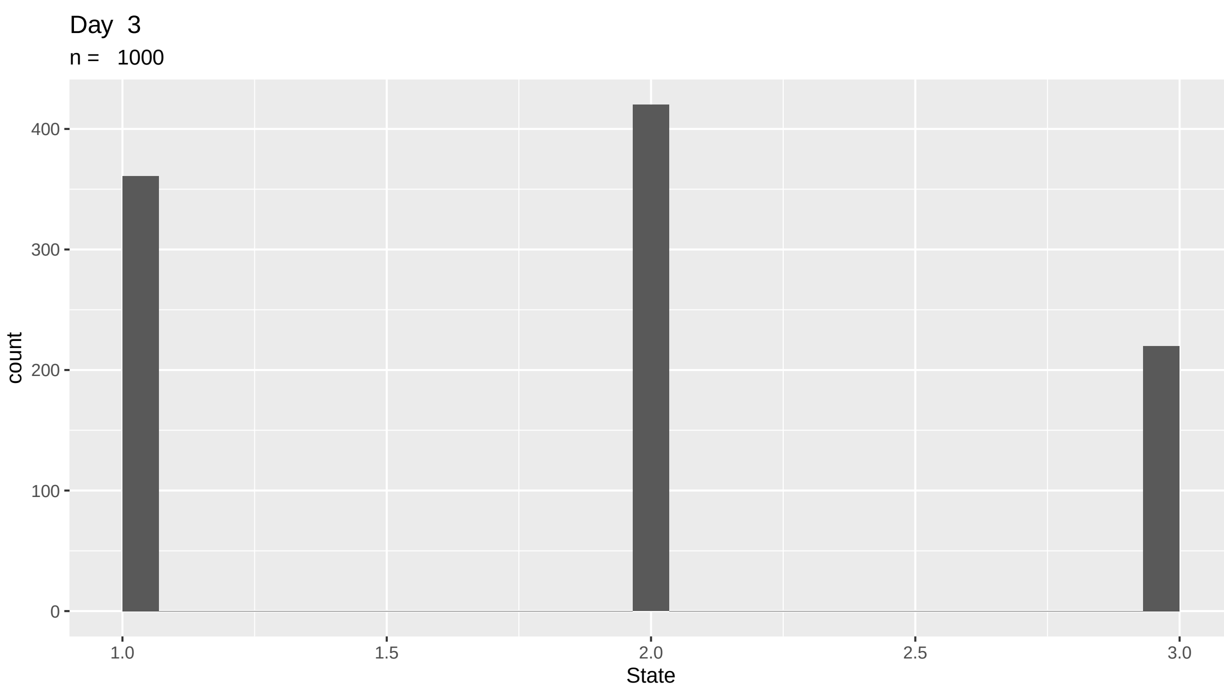
Part b shows the mode given that the previous state at each given day, while part c is showing the expected value given that the previous state at each given day.

Part E

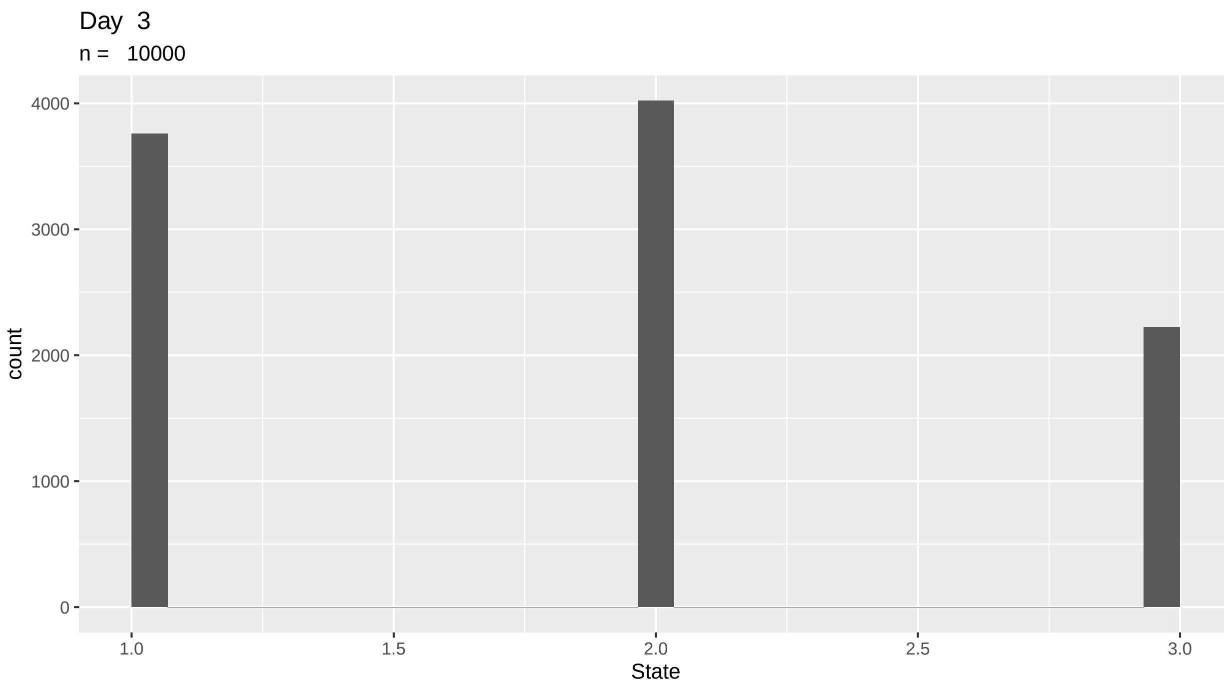
```
set.seed(3149)
P <- matrix(
    data = c(0.45, 0.35, 0.20, 0.36, 0.34, 0.30, 0.25, 0.65, 0.10),
    nrow = 3, ncol = 3, byrow = TRUE
)

# Cumulative Probability
CP <- t(apply(P, 1, cumsum))

repeater <- function(sim_size) {
    X <- matrix(0, sim_size + 1, 1)
    X[1, 1] <- 1
    for (day in 1:3) {
        for (i in 1:sim_size) {
            u <- runif(1)
            X[i + 1, 1] <- 1 * (u < CP[X[i, 1], 1]) +
                2 * (u < CP[X[i, 1], 2]) * (u > CP[X[i, 1], 1]) +
                3 * (u > CP[X[i, 1], 2])
        }

        if (day == 3) {
            g <- X %>%
                as_tibble() %>%
                ggplot(aes(x = V1)) +
                geom_histogram() +
                labs(
                    title = paste("Day ", day),
                    subtitle = paste("n =  ", sim_size),
                    x = "State"
                )
            print(g)
        }
    }
}
sims <- c(100, 1000, 10000, 100000)
for (index in 1:4) {
    repeater(sims[index])
}
```

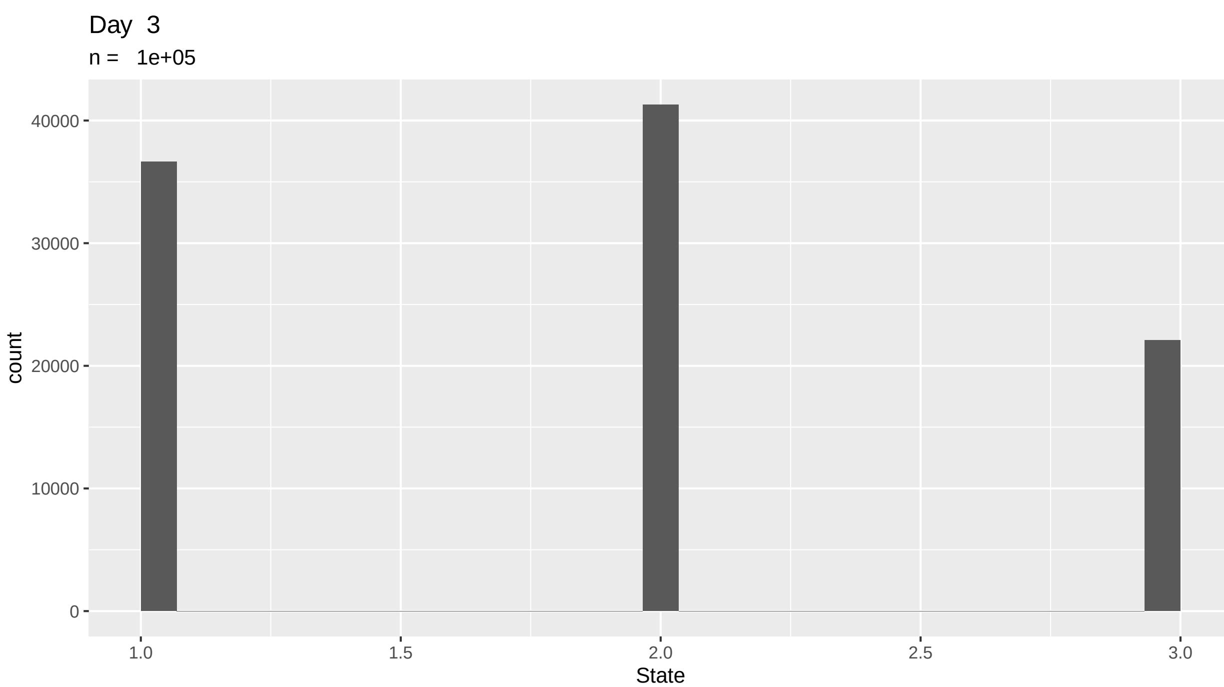


Day 3
n = 100
count
State



Day 3
n = 1000
count
State





$$PDF\left(X_3\right) = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} * \begin{bmatrix} 0.45 & 0.35 & 0.20 \\ 0.50 & 0.25 & 0.25 \\ 0.55 & 0.35 & 0.10 \end{bmatrix}^3$$

$$PDF\left(X_3\right) = \begin{bmatrix} 0.3685687 & 0.4133447 & 0.2180867 \end{bmatrix}$$

From the simulation of $n = 100, 1000, 10^4$ and 10^5, the values converge on the pdf value for X_3 since the accuracy of the models increases as more datapoints are tested.

Part F

```r
## Part F
set.seed(31455034)
P <- matrix(
    data = c(0.45, 0.35, 0.20, 0.36, 0.34, 0.30, 0.25, 0.65, 0.10),
    nrow = 3, ncol = 3, byrow = TRUE
)

# Cumulative Probability
CP <- t(apply(P, 1, cumsum))
markov_proc_del_start <- function(sim_size, starting_day) {
    X <- matrix(1, sim_size + 1, 1)
    for (day in starting_day:3) {
        for (i in 1:sim_size) {
            u <- runif(1)
            X[i, 1] <- 1 * (u < CP[X[i, 1], 1]) +
                2 * (u < CP[X[i, 1], 2]) * (u > CP[X[i, 1], 1]) +
                3 * (u > CP[X[i, 1], 2])
        }
    }
    pi <- matrix(0, 1, 3)
    pi[1, 1] <- sum(X == 1) / sim_size
    pi[1, 2] <- sum(X == 2) / sim_size
    pi[1, 3] <- sum(X == 3) / sim_size
    return(pi)
}

set.seed(31455034)
sims <- c(100, 1000, 10000, 100000)
for (index in 1:4) {
    result <- markov_proc_del_start(sims[index], 2)
    # print(result)
    print(knitr::kable(result, caption = glue("Simulation size: {sims[index]}")))
    # cat("\n")
    cat("\n\n")
    print(glue("Pr(X_3>1|X_1=1) = {result[1,2]} + {result[1,3]} \n"))
    cat("\n\n")
    print(glue("Pr(X_3>1|X_1=1) = {result[1,2]} + {result[1,3]} \n"))
    cat("\n\n")
}
```

Table 5: Simulation size: 100

0.32	0.43	0.26

Pr(X_3>1|X_1=1) = 0.43 + 0.26

Pr(X_3>1|X_1=1) = 0.43 + 0.26

Table 6: Simulation size: 1000

0.393	0.383	0.225

Pr(X_3>1|X_1=1) = 0.383 + 0.225

Pr(X_3>1|X_1=1) = 0.383 + 0.225

Table 7: Simulation size: 10000

0.3718	0.4172	0.2111

Pr(X_3>1|X_1=1) = 0.4172 + 0.2111

Pr(X_3>1|X_1=1) = 0.4172 + 0.2111

Table 8: Simulation size: 1e+05

0.37707	0.40805	0.21489

Pr(X_3>1|X_1=1) = 0.40805 + 0.21489

Pr(X_3>1|X_1=1) = 0.40805 + 0.21489

Table 9: Theoretical Value

0.3785	0.4065	0.215

Pr(X_3>1|X_1=1) = 0.4065 + 0.215

Pr(X_3>1|X_1=1) = 0.4065 + 0.215

The Monte Carlo estimation is done through random sampling to estimate the true probability. However for this to be done accurately, the sampling size needs to be large, since bias in the sample need to minimised. This can be seen in the different values of n that is used to estimate, $Pr\left(X_3 > 1|X_1 = 1\right)$. As n grows so does the accuracy, it converges on the theoretical value.

Part G

```
## Part G
set.seed(31455034)
sims <- c(100, 1000, 10000, 100000)
for (index in 1:4) {
    result <- markov_proc_del_start(sims[index], 3)
    print(knitr::kable(result, caption = glue("Simulation size: {sims[index]}")))
    # cat("\n")
    cat("\n\n")
    print(glue("Pr(X_3>1|X_2=1) = {result[1,2]} + {result[1,3]} \n"))
    cat("\n\n")
    print(glue("Pr(X_3>1|X_2=1) = {result[1,2]} + {result[1,3]} \n"))
    cat("\n\n")
}
```

Table 10: Simulation size: 100

0.44	0.33	0.24

Pr(X_3>1|X_2=1) = 0.33 + 0.24

Pr(X_3>1|X_2=1) = 0.33 + 0.24

Table 11: Simulation size: 1000

0.447	0.36	0.194

Pr(X_3>1|X_2=1) = 0.36 + 0.194

Pr(X_3>1|X_2=1) = 0.36 + 0.194

Table 12: Simulation size: 10000

0.4444	0.3507	0.205

Pr(X_3>1|X_2=1) = 0.3507 + 0.205

Pr(X_3>1|X_2=1) = 0.3507 + 0.205

Table 13: Simulation size: 1e+05

0.44935	0.35178	0.19888

Pr(X_3>1|X_2=1) = 0.35178 + 0.19888

Pr(X_3>1|X_2=1) = 0.35178 + 0.19888

Table 14: Theoretical Value

0.45	0.35	0.2

Pr(X_3>1|X_2=1) = 0.35 + 0.2

Pr(X_3>1|X_2=1) = 0.35 + 0.2

Markov processes are memoryless hence, $Pr\left(X_3 > 1|X_1 = 1, X_2 = 1\right) = Pr\left(X_3 > 1|X_2 = 1\right)$. The Monte Carlo estimation is done through random sampling to estimate the true probability. However for this to be done accurately, the sampling size needs to be large, since bias in the sample need to minimised. This can be seen in the different values of n that is used to estimate, $Pr\left(X_3 > 1|X_2 = 1\right)$. As n grows so does the accuracy, it converges on the theoretical value.

Part H

The difference between the two questions is that (f) is checking the probability of day 3 not being sunny given that it was sunny on day 1 which means we should be calculating (or rather simulating) for 2 days. (g) on the other hand, is checking the same idea except it tells us that day 1 & 2 are both sunny. Now since we know that we are working with a Markov process, we just need to know the most recent event as a Markov process can be calculated from only one starting point without any past data. Therefore we should simulating for only 1 day and perform calculations from day 2 $\rightarrow$ 3.

Question 2

Part A

```r
data_sheet1 <- read_excel("Data.xlsx", sheet = 1) %>%
    select(Dataset1, Dataset2)

data_sheet1 <- data_sheet1[-c(1), ]

data_sheet1 <- transform(
    data_sheet1,
    Dataset1 = as.numeric(Dataset1),
    Dataset2 = as.numeric(Dataset2)
)

count_matrix1 <- matrix(0, 3, 3)

# Creating Count
for (index in 2:length(data_sheet1$Dataset1)) {
    row <- data_sheet1$Dataset1[index - 1]
    col <- data_sheet1$Dataset1[index]
    count_matrix1[row, col] <- count_matrix1[row, col] + 1
}

# Starting state is 1 and first datapoint is 1
count_matrix1[1, 1] <- count_matrix1[1, 1] + 1

# Creating transition
tran_mat1 <- matrix(
    c(
        count_matrix1[1, ] / sum(count_matrix1[1, ]),
        count_matrix1[2, ] / sum(count_matrix1[2, ]),
        count_matrix1[3, ] / sum(count_matrix1[3, ])
    ), 3, 3,
    byrow = TRUE
)
```

Table 15: Transition Matrix for Dataset 1

0.5135135	0.4594595	0.0270270
0.0000000	0.3888889	0.6111111
0.6666667	0.1851852	0.1481481

Part B

```r
count_matrix2 <- matrix(0, 3, 3)

# Creating Count
for (index in 2:length(data_sheet1$Dataset2)) {
    row <- data_sheet1$Dataset2[index - 1]
    col <- data_sheet1$Dataset2[index]
    count_matrix2[row, col] <- count_matrix2[row, col] + 1
}

# Creating transition
tran_mat2 <- matrix(c(
```

```
    count_matrix2[1, ] / sum(count_matrix2[1, ]),
    count_matrix2[2, ] / sum(count_matrix2[2, ]),
    count_matrix2[3, ] / sum(count_matrix2[3, ])
), 3, 3, byrow = TRUE)
```

Table 16: Transition Matrix for Dataset 2

0.3157895	0.4210526	0.2631579
0.1666667	0.0416667	0.7916667
0.5945946	0.1621622	0.2432432

Part C

```
knitr::kable((tran_mat1 %^% 10), caption = "DS1 Transition Matrix in 10 periods")
```

Transition Matrix of Dataset 1 in 10 periods

Table 17: DS1 Transition Matrix in 10 periods

0.3699757	0.3600365	0.2699878
0.3699786	0.3599607	0.2700607
0.3700618	0.3600023	0.2699359

```
knitr::kable((tran_mat2 %^% 10), caption = "DS2 Transition Matrix in 10 periods")
```

Transition Matrix of Dataset 2 in 10 periods

Table 18: DS2 Transition Matrix in 10 periods

0.3866172	0.2340858	0.379297
0.3866169	0.2340161	0.379367
0.3866736	0.2340515	0.379275

Part D

Both Dataset 1 and Dataset 2's distribution matrices stablises to 4 decimal places by the time the period approaches 10. In Dataset2, the columns of the transtion matrix converge on the same value, i.e. $Pr(X_{10} = 1|X_1 = 1) = Pr(X_{10} = 1|X_1 = 2) = Pr(X_{10} = 1|X_1 = 3)$ and so one for the other 2 states as well. This has not happened to Dataset1 at period 10 but proper stability occurs after 8 more periods.

Question 3

Part A

```
data_sheet2 <- read_excel("Data.xlsx", sheet = 2) %>%
    select(Dataset1, Dataset2)

data_sheet2 <- data_sheet2[-c(1), ]
states <- tibble(state = rep(1:2, times = length(data_sheet2$Dataset1) / 2 + 1))
states <- states[-c(length(data_sheet2$Dataset1) + 1), ]
```

```
data_sheet2 <- data_sheet2 %>% cbind(states)
data_sheet2 <- transform(
    data_sheet2,
    Dataset1 = as.numeric(Dataset1),
    Dataset2 = as.numeric(Dataset2)
)
data_sheet2 <- data_sheet2 %>% as_tibble()


lambda_states1 <- data_sheet2 %>%
    select(Dataset1, state) %>%
    group_by(state) %>%
    summarise(lambda = 1 / mean(Dataset1))

knitr::kable(lambda_states1, caption = "Lambda States DS1")
```

Table 19: Lambda States DS1

state	lambda
1	0.0240084
2	0.7039102

```
# Make transition intensity matrix
int_matrix1 <- matrix(
    c(
        -lambda_states1$lambda[1], lambda_states1$lambda[1],
        lambda_states1$lambda[2], -lambda_states1$lambda[2]
    ),
    2, 2, TRUE
)
knitr::kable(expm(int_matrix1), caption = "Transition Intentisy Matrix DS1")
```

Table 20: Transition Intentisy Matrix DS1

0.9829453	0.0170547
0.5000322	0.4999678

Part B

```
lambda_states2 <- data_sheet2 %>%
    select(Dataset2, state) %>%
    group_by(state) %>%
    summarise(lambda = 1 / mean(Dataset2))

knitr::kable(lambda_states2, caption = "Lambda States DS2")
```

Table 21: Lambda States DS2

state	lambda
1	0.2732027

state	lambda
2	0.5085200

```r
# Make transition intensity matrix
int_matrix2 <- matrix(
    c(
        -lambda_states2$lambda[1], lambda_states2$lambda[1],
        lambda_states2$lambda[2], -lambda_states2$lambda[2]
    ),
    2, 2, TRUE
)
knitr::kable(expm(int_matrix2), caption = "Transition Intentisy Matrix DS2")
```

Table 22: Transition Intentisy Matrix DS2

0.8104436	0.1895564
0.3528266	0.6471734

Part C

```r
knitr::kable(expm(int_matrix1) %^% 10, caption = "DS1 Transition Matrix in 10 jumps")
```

Table 23: DS1 Transition Matrix in 10 jumps

0.9670404	0.0329596
0.9663507	0.0336493

```r
knitr::kable(expm(int_matrix2) %^% 10, caption = "DS2 Transition Matrix in 10 jumps")
```

Table 24: DS2 Transition Matrix in 10 jumps

0.6506527	0.3493473
0.6502500	0.3497500

Part D

Dataset 1 's distribution matrix stablises to 2 decimal places by the time the period approaches 10. Dataset 2's distribution matrix stablises to 3 decimal places by the time the period approaches 10. Also for Dataset 1 and Dataset 2, the columns of the transtion matrix converge on the same value, i.e. $Pr(X_{10} = 1|X_1 = 1) = Pr(X_{10} = 1|X_1 = 2) = Pr(X_{10} = 1|X_1 = 3)$ and so one for the other 2 states as well.