# STAT 440 Homework 12

Charlie Lu (Cxl5159)

October 2022

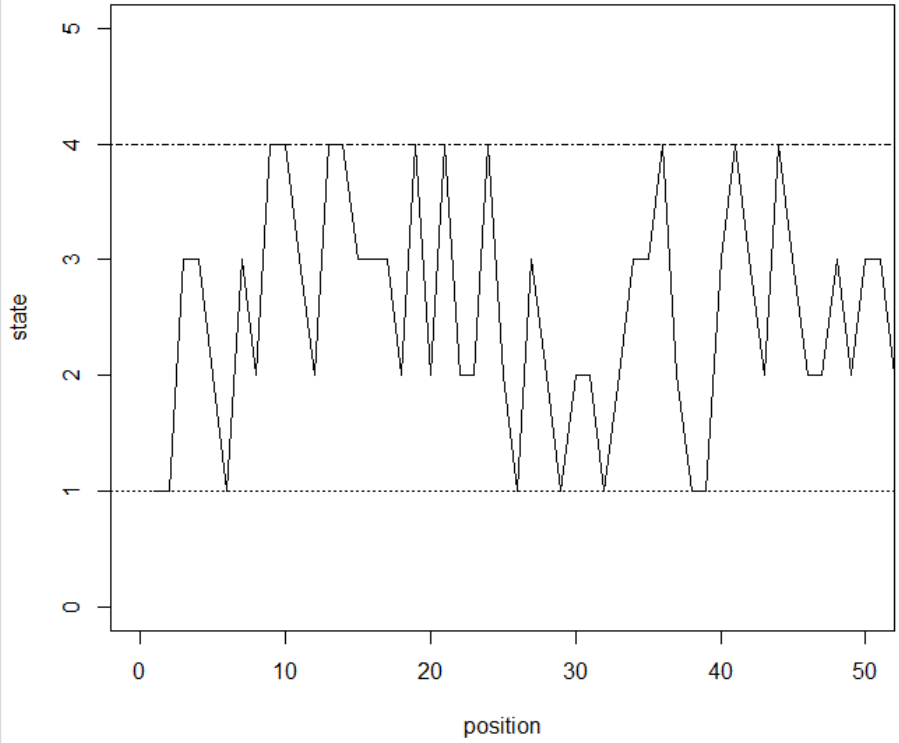
## 1 A

```
4  P <- t(matrix(c(0.180,0.274,0.426,0.120,
5                  0.171,0.367,0.274,0.188,
6                  0.161,0.339,0.375,0.125,
7                  0.079,0.355,0.384,0.182),nrow=4, ncol=4))
8  P
9  P%^%5
10 P%^%50
11 P%^%500
12
13:1  (Top Level)

Console  Terminal  Background Jobs
R 4.2.1 · ~/
> P
      [,1]  [,2]  [,3]  [,4]
[1,] 0.180 0.274 0.426 0.120
[2,] 0.171 0.367 0.274 0.188
[3,] 0.161 0.339 0.375 0.125
[4,] 0.079 0.355 0.384 0.182
> P%^%5
          [,1]      [,2]      [,3]      [,4]
[1,] 0.1546840 0.3409589 0.3498497 0.1545074
[2,] 0.1546760 0.3409679 0.3498378 0.1545183
[3,] 0.1546805 0.3409631 0.3498445 0.1545119
[4,] 0.1546730 0.3409700 0.3498363 0.1545207
> P%^%50
          [,1]      [,2]      [,3]      [,4]
[1,] 0.1546783 0.3409652 0.3498417 0.1545148
[2,] 0.1546783 0.3409652 0.3498417 0.1545148
[3,] 0.1546783 0.3409652 0.3498417 0.1545148
[4,] 0.1546783 0.3409652 0.3498417 0.1545148
> P%^%500
          [,1]      [,2]      [,3]      [,4]
[1,] 0.1546783 0.3409652 0.3498417 0.1545148
[2,] 0.1546783 0.3409652 0.3498417 0.1545148
[3,] 0.1546783 0.3409652 0.3498417 0.1545148
[4,] 0.1546783 0.3409652 0.3498417 0.1545148
>
```

## 2 B

```
#markov chain
set.seed(440)
# simulate discrete Markov chains according to transition matrix P
run.mc.sim <- function( P, num.iters = 100000 ) {

  # number of possible states
  num.states <- nrow(P)

  # stores the states X_t through time
  states     <- numeric(num.iters)

  # initialize variable for first state
  states[1]    <- 1

  for(t in 2:num.iters) {

    # probability vector to simulate next state X_{t+1}
    p  <- P[states[t-1], ]

    # draw from multinomial and determine state
    states[t] <-  which(rmultinom(1, 1, p) == 1)
  }
  return(states)
}

num.chains     <- 1
num.iterations <- 100000

chain.states  <- matrix(NA, ncol=num.chains, nrow=num.iterations)
set.seed(440)
# simulate chains
for(c in seq_len(num.chains)){
  chain.states[,c] <- run.mc.sim(P)
}

matplot(chain.states, type='l', lty=1, col=1:5, xlim=c(0, 50), ylim
abline(h=1, lty=3)
abline(h=4, lty=4)
```

# 3 B



## 4 C

```
52 #proportions
53 temp <- unlist(as.list(chain.states))
54 A <- length(which(temp==1))/(length(temp))
55 C <- length(which(temp==2))/(length(temp))
56 G <- length(which(temp==3))/(length(temp))
57 t <- length(which(temp==4))/(length(temp))
58 proportions <- c(A,C,G,t)
59 proportions
60
```

60:1 (Top Level)

Console Terminal × Background Jobs ×

R 4.2.1 · ~/

```
> #proportions
> temp <- unlist(as.list(chain.states))
> A <- length(which(temp==1))/(length(temp))
> C <- length(which(temp==2))/(length(temp))
> G <- length(which(temp==3))/(length(temp))
> t <- length(which(temp==4))/(length(temp))
> proportions <- c(A,C,G,t)
> proportions
[1] 0.15376 0.34285 0.34991 0.15348
>
```

## 5 D

```
61  #sequence counting
62  for(i in 1:length(temp)){
63    if (temp[i] == 2 & temp[i+1] == 3){
64      count = count + 1
65    }
66  }
67  prop <- count/(length(temp)-1)
68  prop
69
70
```

71:1 (Top Level)

Console | Terminal | Background Jobs

R 4.2.1 · ~/

```
> count = 0
> #sequence counting
> for(i in 1:length(temp)){
+   if (temp[i] == 2 & temp[i+1] == 3){
+     count = count + 1
+   }
+ }
> prop <- count/(length(temp)-1)
> prop
[1] 0.09528095
```

# 6 E

```
73  #initial prob
74  P
75  proportion <- 0.274*length(which(temp==2))/length(temp)
76  proportion
77
78
```

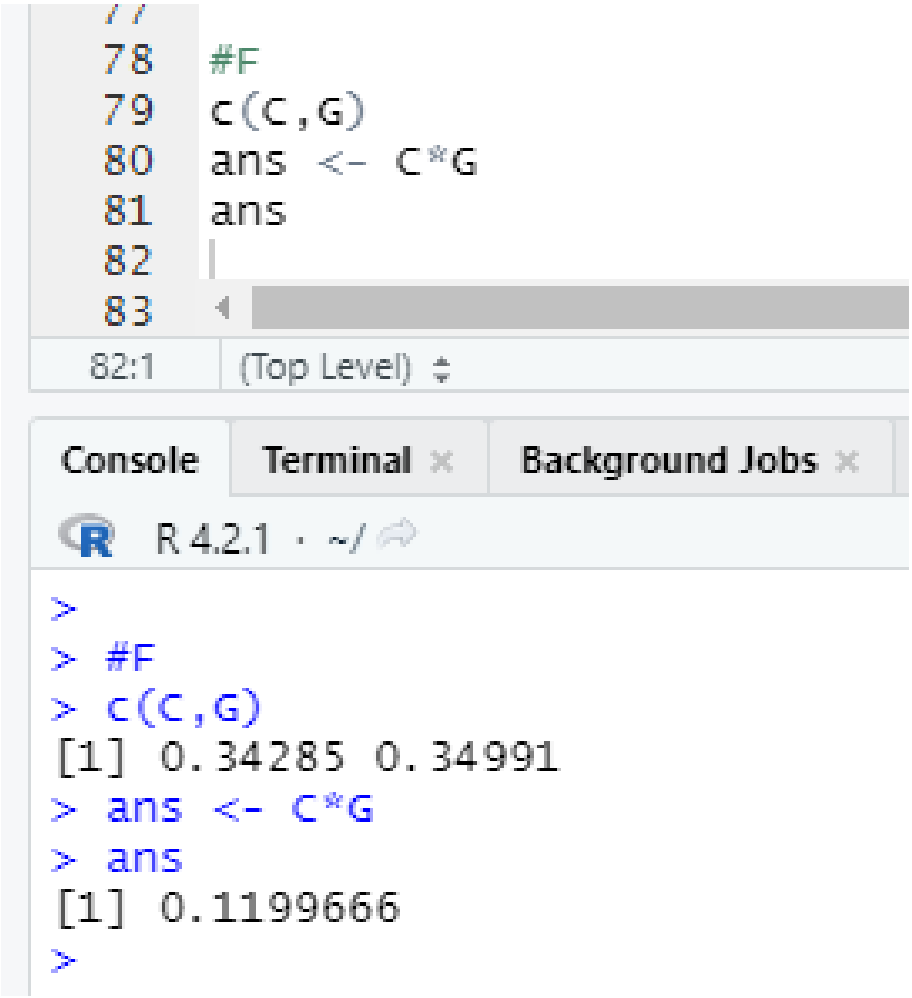
78:1 (Top Level)

Console Terminal Background Jobs

R 4.2.1 · ~/

```
> proportion <- 0.274*length(which(temp==2))/length(temp)
> proportion
[1] 0.0939409
>
```

## 7 F1


78 #F
79 c(C,G)
80 ans <- C*G
81 ans
82
83
82:1 (Top Level)
Console
Terminal
Background Jobs
R 4.2.1 · ~/
> #F
> c(C,G)
[1] 0.34285 0.34991
> ans <- C*G
> ans
[1] 0.1199666
>