MATH50010 Coursework

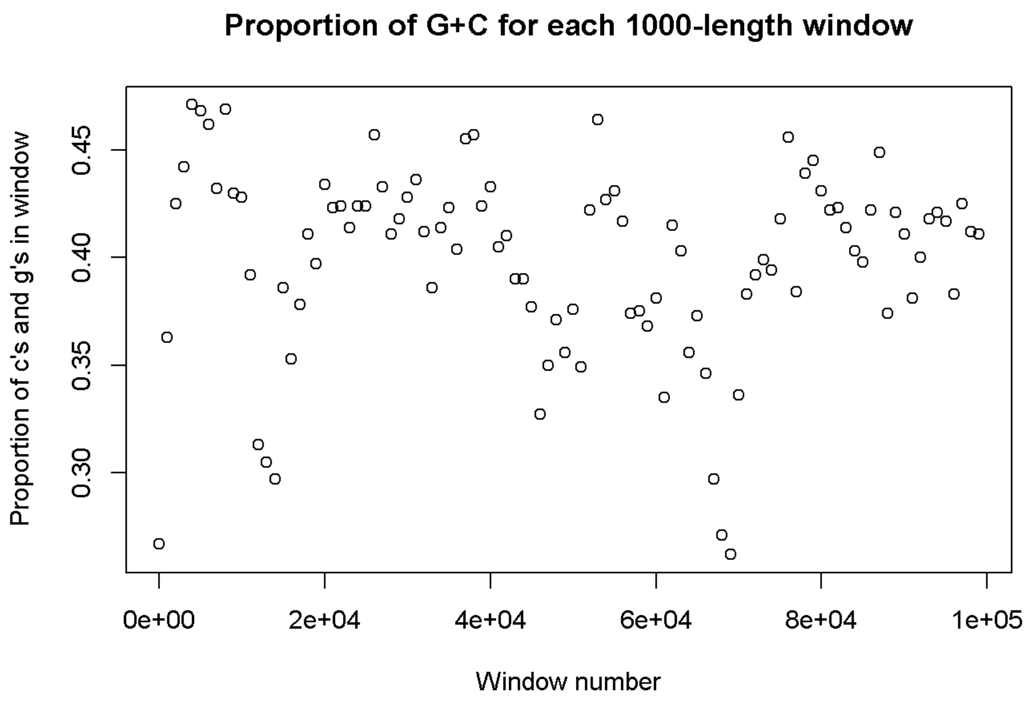
Joshua Tegegne 01705625

17/12/2020

```r
library(seqinr)
library(comprehenr)
library(gtools)
#importing functions that we will use to work with our sequence, such as comprehensions, vectors and matrice
s
dat<-read.fasta("sequences.fa")
data<- dat[[1]]
#data is the sequence we will be investigating
letterprops <- count(data, 1)/length(data)
prop_a <- letterprops["a"]
prop_c <- letterprops["c"]
prop_g <- letterprops["g"]
prop_t <- letterprops["t"]
#We find the number of occurrences of each letter, then divide by the length of our sequence to get the prop
ortions
#
```

1. Proportions of a, c, g, t respectively are 0.3052582,0.1995605,0.2005196,0.2946616

```r
num = seq(0,99000,1000)
#this vector contains the number at the start of every 1000-letter window we will inspect
props = to_vec(for(i in num) (count(data[i:(i+1000)],1)["g"]+count(data[i:(i+1000)],1)["c"])/1000 )
#To get the proportions of G+C we iterate over each window from i to i+1000, calculating the number of occur
rences of g and c and dividing by 1000
plot(num, props, xlab = "Window number", ylab = "Proportion of c's and g's in window", main= "Proportion of
G+C for each 1000-length window")
```



2. Above we find the proportion of G+C from position 1 to 1000, then 1000 to 2000, and so on.

```r
letterl <- permutations(n=4,r=2,v= c("a","g","c","t"),repeats.allowed = TRUE)
#List of the 16 permutations of a,c,g and t with length 2
q_ij <- to_vec(for (i in 1:16) letterprops[letterl[i,1]]*letterprops[letterl[i,2]] )
#iterate over every letter pair, and calculate n_ij/(n-1)
o <- data.frame(letterl,q_ij)
names(o) <- c("i","j","q_ij")
print (o)
```

```
##    i j        q_ij
## 1  a a 0.09318260
## 2  a c 0.06091749
## 3  a g 0.06121026
## 4  a t 0.08994790
## 5  c a 0.06091749
## 6  c c 0.03982440
## 7  c g 0.04001579
## 8  c t 0.05880283
## 9  g a 0.06121026
## 10 g c 0.04001579
## 11 g g 0.04020811
## 12 g t 0.05908544
## 13 t a 0.08994790
## 14 t c 0.05880283
## 15 t g 0.05908544
## 16 t t 0.08682549
```

3a) Above is the table representing the value of q_{ij}, using the equation $q_{ij} = p_i p_j$, where p_k is the proportion of letter k in the sequence that we worked out in Q1

```
doubles <- to_vec(for(i in 1:16) paste(letterl[i,1],letterl[i,2], sep = ""))
#This is a vector of the 16 permutations but instead of "a", "c" for example,we have "ac"
likelihood_val <- sum(to_vec(for(i in 1:16) count(data,2)[doubles[i]]*log( count(data,2)[doubles[i]]/(q_ij[i
]*(length(data)-1)) )   ))
#we find the sum of n_ij *log(nij/((n-1)*q_ij) over the 16 possible pairs of i,j
print (likelihood_val)
```

```
## [1] 15368.99
```

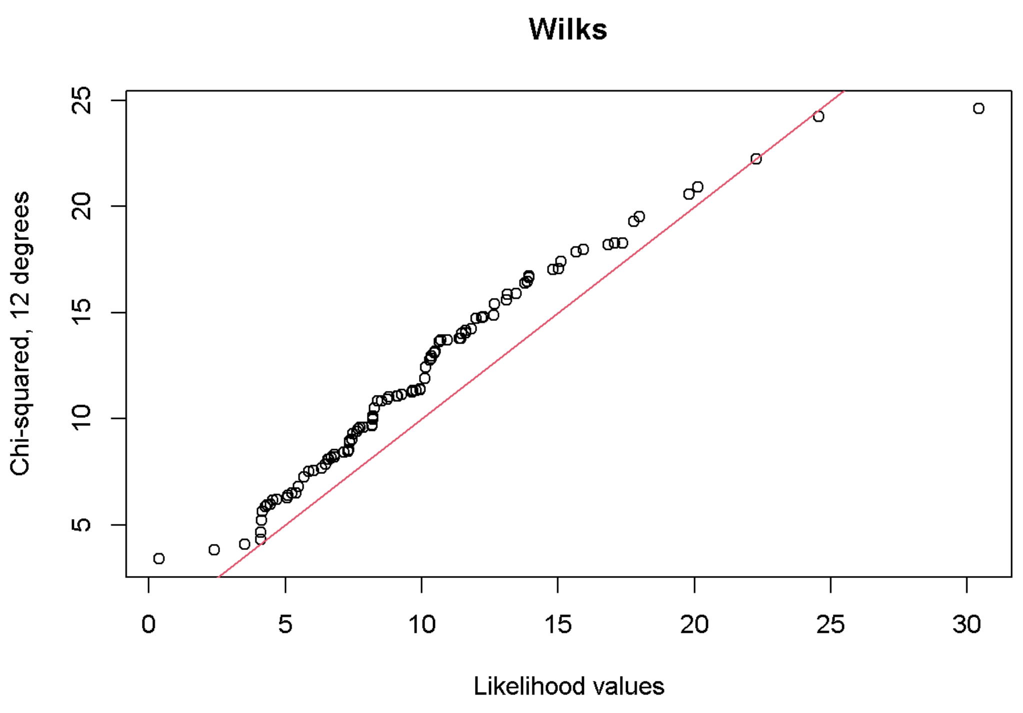
3b) For the original data our log likelihood ratio is 1.536898910^{4}

```
likelihood_vals <- matrix(nrow = 1, ncol = 500)
for(z in 1:100) {
  seq_sample <- permutation(data)
  vals <- count(seq_sample,2)
  likelihood_vals[,z ] <- sum(to_vec(for(i in 1:16) vals[doubles[i]]*log( vals[doubles[i]]/(q_ij[i]*(length(
data)-1)) )   ))
}

#In our 10 by 16 matrix, the i j'th element is (p_j/q_j)^n_j for the i'th sample, where j is the j'th permuta
tion of #acgt that is length 2.
#We then compute our likelihood ratios by taking each column, and taking the log of the product of the eleme
nt
print (likelihood_vals[1:100])
```

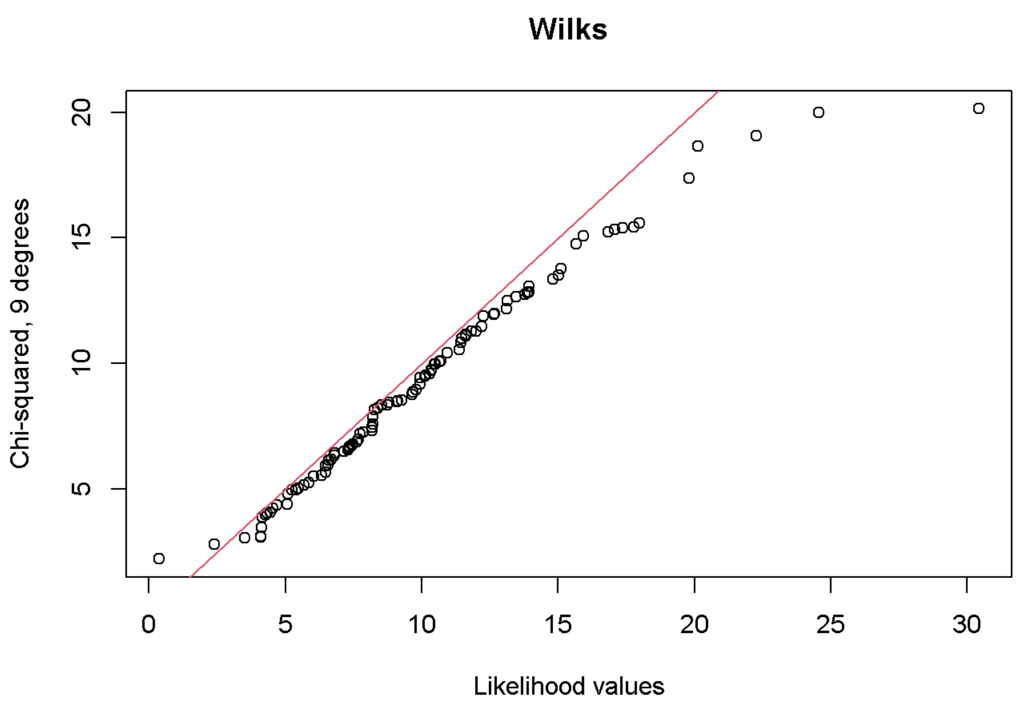
```
##   [1]  5.6909107  4.3694462  7.9560042  8.9880643  5.2294814  2.6198580
##   [7]  4.5639123  8.6814005  6.8967862  2.5332742  3.3976664  3.2859180
##  [13]  7.8240243  2.9287411  5.7338492  3.2917375  4.4007529  6.9615055
##  [19]  5.9148882  4.1053775 12.2711966  5.3203726  1.7473015  4.8181781
##  [25]  4.1079497  4.6264984  7.5528308  2.2263228  3.6803911  3.7171740
##  [31]  3.9264797 15.2119457  5.1482363  5.4745233  7.4090958  2.7338967
##  [37]  5.7128316  6.5543491  4.1908108  3.2427449  5.1697264  5.8121609
##  [43]  5.0510523  2.1321539  3.0237720  3.3459309  2.8438788  4.0854062
##  [49]  3.8070420  3.5806457  8.5423302  3.6501008  5.0696825  4.9714885
##  [55]  4.0836963  7.5051570  3.4040373  8.8802763  6.3171110  6.9588651
##  [61]  6.3346538  6.0999699  2.6912594  6.5612834  2.1589621  0.1824472
##  [67]  5.8003475  3.6565289  2.2703721  6.0028508  5.2519926  8.4217289
##  [73]  4.8431858  2.3426595  3.6712491  2.0421292  4.8989927  3.8344778
##  [79]  3.1592756 11.1295859  4.1021609  4.1316098  2.0480678  3.7370490
##  [85]  2.5496762  4.2617279  1.1943328  6.1211067  5.1768732  5.3491595
##  [91]  6.7344431  2.0599301  9.8964896  4.9646979 10.0593785  6.9341470
##  [97]  3.2371180  2.0774668  4.5479792  3.8729832
```

3c) Above we calculate the log likelihood ratio for 100 different samples. All ratios are greater than 1, but for the sample data the likelihoods are at 10 , whereas the likelihood is over 15000 for the null model so we can reject the null hypothesis comfortably.

```
qqplot(2*likelihood_vals, rchisq(100,df=12), xlab = "Likelihood values", ylab="Chi-squared, 12 degrees", mai
n = "Wilks")
abline(0,1,col=2)
```



```
qqplot(2*likelihood_vals, rchisq(100,df=9),xlab = "Likelihood values", ylab="Chi-squared, 9 degrees", main =
"Wilks")
abline(0,1,col=2)
```



3c) There are 16 different

transition probabilities and 4 possible states. 16-4 = 12 we try 12 degrees but it seems like 9 degrees is the most accurate. This may be because of our constraints. As $\sum_j p_{ij} = 1$, there are 4 parameters that depend on the other 12 parameters, so we have 12 free parameters under the alternative model. Similarly, we have 3 = 4-1 free parameters under the null model. 12-3 = 9, this number of degrees supports Wilks' theorem. The points that are far away from the y=x line generally have a higher likelihood, so there may be some rounding error miscalculations that affect larger values the most.

4a) $Pr(X_n = x_n, .., X_0 = x_0) = Pr(X_n = x_n | X_{n-1} = x_{n-1}, .., X_0 = x_0) Pr(X_{n-1} = x_{n-1}, .., X_0 = x_0)$
$= Pr(X_n = x_n | X_{n-1} = x_{n-1}) Pr(X_{n-1} = x_{n-1}, .., X_0 = x_0) =$. Note that we use the Markov property and conditional probability.

$Pr(X_n = x_n|X_{n-1} = x_{n-1})Pr(X_{n-1} = x_{n-1}, .., X_0 = x_0) = Pr(X_0 = x_0)\prod Pr(X_i = x_i|X_{i-1} = x_{i-1})$ $\implies L(P) = Pr(X_0 = x_0)\prod_{i=1}^{n} p_{x_{i-1}x_i} \implies Pr(X_0 = x_0)\prod_{i,j\in\epsilon} p_{ij}^{n_{ij}}$ Taking logs we get $\log(L(P)) = log(Pr(X_0 = x_0)) + \prod_{i,j\in\epsilon} n_{ij}\log(p_{ij})$

4b) If we differentiate the log likelihood Of P we get n_{ij}/p_{ij}, but we can't do anything else with this. Instead we will solve using Lagrange multipliers. For every state we have a Lagrange multiplier n_i, and our constraint equations are $n_i(\sum_j p_{ij} - 1)$. We seek to find the stationary points of the function $Log(L(P)) - (\sum_i n_i)(\sum_j p_{ij} - 1))$ Differentiate partially with respect to n_i and we get $\frac{n_{ij}}{p_{ij}} - n_i = 0 \implies \sum_j p_{ij} = \sum_j \frac{n_{ij}}{n_i} = 1 \implies n_i = \sum_j n_{ij}$ Therefore our MLE is $\frac{n_{ij}}{n_i}$

```
p_hat <-  matrix(nrow = 4, ncol = 4)
# initialise our markov matrix where the i j'th element will be the MLE of the transition probability from t
he i'th state to the j'th state. We say the states are in alphabetical order: a, c, g, t.
n_i <- to_vec(for(i in seq(1,13,4)) sum(count(data,2)[i:(i+3)]) )
#vector of length 4 where elements are n_a, n_c, n_g, n_t
states <- c("a","c","g","t")
for (i in 1:4) {
  for (j in 1:4) {
      p_hat[i,j] = count(data, 2)[paste(states[i],states[j],sep = "")]/n_i[i]
  }
}
#iterating over every postion in our matrix and working out p_ij
colnames(p_hat) <- c("a","c","g","t")
rownames(p_hat) <- c("a","c","g","t")
print (p_hat)
```

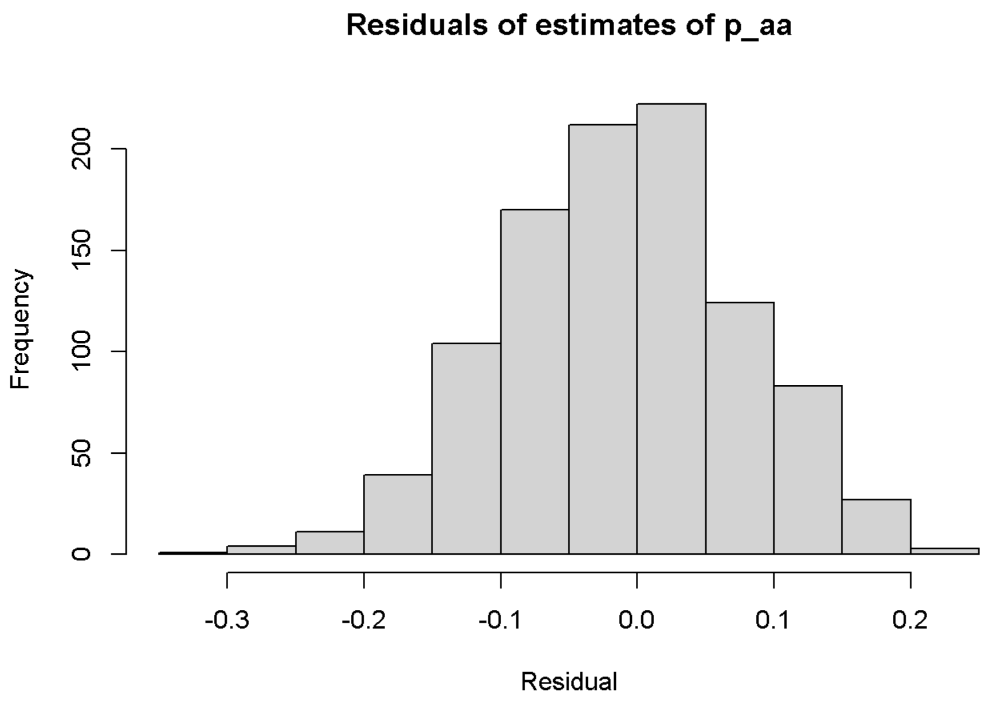
```
##           a         c         g         t
## a 0.3952795 0.2036908 0.1922869 0.2087428
## c 0.3256384 0.2251043 0.1641480 0.2851093
## g 0.2497511 0.2268376 0.2245408 0.2988705
## t 0.2359712 0.1594204 0.2173355 0.3872730
```

5a) Above is the transition matrix

```
n<- 1000 # number of realisations
m<-100 # length of chain to be simulated
realisations <- matrix(nrow = n, ncol = m)
# matrix where each column represents an iteration

for (z in 1:n){
  x<-vector(length=100)
  x[1]<-sample(x=4,size=1)
  for(i in 2:m){
    x[i]<-sample(x=4,size=1,prob=p_hat[x[i-1],])
  }
  realisations[z,] = x
}
# 1, 2, 3, 4 represent "a", "c", "g" and "t" respectively
```

5a) Above is an example of n iterations of m-length chains

```r
n <- 1000
estimations <- matrix(nrow = n,ncol = 16)
# Each row has 16 elements containing the estimated transition probabilities
for (i in 1:n){
  samp <- gsub(1,"a", realisations[i,])
  samp <- gsub(2, "c", samp)
  samp <- gsub(3, "g", samp)
  samp <- gsub(4, "t", samp)
  samp <- s2c(paste(samp, collapse = ''))
  # replacing numbers with letters and joining the letters together to convert into a chain we can work with
  for (j in 1:16){
    prop <- count(samp,2)
    if (j<5){    #In this loop we work out the MLE transition probabilities for each iteration and add to our
matrix
      estimations[i,j] = prop[doubles[j]]/sum( to_vec( for(k in 1:4) prop[doubles[k]]) )
    }
    if (j>4 & j<9){
      estimations[i,j] = prop[doubles[j]]/sum( to_vec( for(k in 5:9) prop[doubles[k]]) )
    }
    if (j>8 & j<13){
      estimations[i,j] = prop[doubles[j]]/sum( to_vec( for(k in 9:12) prop[doubles[k]]) )
    }
    if (j>12){
      estimations[i,j] = prop[doubles[j]]/sum( to_vec( for(k in 13:16) prop[doubles[k]]) )
    }

    #We work out the 16 estimated transition probabilities for each iteration
  }

}
hist(estimations[,1]-p_hat[1,1], xlab = "Residual", main= "Residuals of estimates of p_aa")
```



```r
#Normalise by taking away the expected value of p_aa vs the estimates, and plot a histogram
```

5b) The histogram above shows that the $p_a a$ estimators, the probability of staying on state a, is somewhat unbiased as the residuals seem to generally follow a normal distribution, the expected value is close to 0

```r
cor(estimations)
```

```
##             [,1]         [,2]         [,3]         [,4]         [,5]
##  [1,]  1.000000000 -0.367626904 -0.366550890 -0.405990531  0.064719111
##  [2,] -0.367626904  1.000000000 -0.329661988 -0.297425944  0.062037974
##  [3,] -0.366550890 -0.329661988  1.000000000 -0.221655356 -0.108035254
##  [4,] -0.405990531 -0.297425944 -0.221655356  1.000000000 -0.032750036
##  [5,]  0.064719111  0.062037974 -0.108035254 -0.032750036  1.000000000
##  [6,] -0.021830008  0.104975530 -0.071516149 -0.014017296 -0.195462839
##  [7,] -0.013777786  0.044010194 -0.045952356  0.015279661 -0.305782168
##  [8,] -0.038008083  0.150143983 -0.107598130 -0.006956493 -0.344352021
##  [9,]  0.038450449 -0.013553457  0.001774416 -0.031936617 -0.225931286
## [10,]  0.003968148 -0.004368180  0.003797545 -0.003762707  0.119690652
## [11,] -0.035384062  0.009585910  0.043671075 -0.012452930  0.008532122
## [12,] -0.008768840  0.008330577 -0.043929185  0.044896945  0.099987051
## [13,]  0.031953567  0.042162338 -0.030975055 -0.050409264  0.020490864
## [14,]  0.032579620  0.033455486 -0.051695548 -0.021483046  0.077146962
## [15,] -0.013682676 -0.088414084  0.085205155  0.024094351 -0.092920166
## [16,] -0.040667170  0.014603898 -0.008076260  0.039635731  0.005035259
##             [,6]        [,7]         [,8]         [,9]        [,10]
##  [1,] -0.02183001 -0.01377779 -0.038008083  0.038450449  0.003968148
##  [2,]  0.10497553  0.04401019  0.150143983 -0.013553457 -0.004368180
##  [3,] -0.07151615 -0.04595236 -0.107598130  0.001774416  0.003797545
##  [4,] -0.01401730  0.01527966 -0.006956493 -0.031936617 -0.003762707
##  [5,] -0.19546284 -0.30578217 -0.344352021 -0.225931286  0.119690652
##  [6,]  1.00000000 -0.18493165 -0.166185677 -0.151301115  0.098867819
##  [7,] -0.18493165  1.00000000 -0.282971242 -0.169898896  0.095361497
##  [8,] -0.16618568 -0.28297124  1.000000000 -0.184904516  0.122296352
##  [9,] -0.15130112 -0.16989890 -0.184904516  1.000000000 -0.333216282
## [10,]  0.09886782  0.09536150  0.122296352 -0.333216282  1.000000000
## [11,] -0.01872339  0.05388211 -0.034171467 -0.294293967 -0.274069367
## [12,]  0.07183094  0.02817619  0.096495628 -0.391504664 -0.339195455
## [13,]  0.02699673 -0.01020420  0.024212236 -0.023224087  0.016539104
## [14,]  0.11817653  0.06966791  0.011229961  0.028700134 -0.053475909
## [15,] -0.11062959 -0.02774089 -0.064228743 -0.004665422  0.066107032
## [16,] -0.01596288 -0.01890623  0.026041107  0.003136917 -0.032123236
##             [,11]        [,12]       [,13]        [,14]        [,15]
##  [1,] -0.035384062 -0.008768840  0.03195357  0.032579620 -0.013682676
##  [2,]  0.009585910  0.008330577  0.04216234  0.033455486 -0.088414084
##  [3,]  0.043671075 -0.043929185 -0.03097506 -0.051695548  0.085205155
##  [4,] -0.012452930  0.044896945 -0.05040926 -0.021483046  0.024094351
##  [5,]  0.008532122  0.099987051  0.02049086  0.077146962 -0.092920166
##  [6,] -0.018723387  0.071830943  0.02699673  0.118176529 -0.110629585
##  [7,]  0.053882106  0.028176192 -0.01020420  0.069667907 -0.027740894
##  [8,] -0.034171467  0.096495628  0.02421224  0.011229961 -0.064228743
##  [9,] -0.294293967 -0.391504664 -0.02322409  0.028700134 -0.004665422
## [10,] -0.274069367 -0.339195455  0.01653910 -0.053475909  0.066107032
## [11,]  1.000000000 -0.361610449  0.02706470  0.009740265 -0.040297498
## [12,] -0.361610449  1.000000000 -0.01681383  0.012137318 -0.019256821
## [13,]  0.027064695 -0.016813827  1.00000000 -0.256349533 -0.247363520
## [14,]  0.009740265  0.012137318 -0.25634953  1.000000000 -0.289108453
## [15,] -0.040297498 -0.019256821 -0.24736352 -0.289108453  1.000000000
## [16,]  0.003836950  0.022523586 -0.47697807 -0.269102180 -0.434601352
##             [,16]
##  [1,] -0.040667170
##  [2,]  0.014603898
##  [3,] -0.008076260
##  [4,]  0.039635731
##  [5,]  0.005035259
##  [6,] -0.015962876
##  [7,] -0.018906229
##  [8,]  0.026041107
##  [9,]  0.003136917
## [10,] -0.032123236
## [11,]  0.003836950
## [12,]  0.022523586
## [13,] -0.476978074
## [14,] -0.269102180
## [15,] -0.434601352
## [16,]  1.000000000
```

5c) Above is the 16 by 16 correlation matrix for our transition probabilities, it seems like there is weak/no correlation between most pairs because the non-diagonal elements are generally not close to 0

6. The Markov model is limited in its capabilities because we can only investigate where 2-length chain occurs (a jump from one state to another). For example we can’t investigate where trimers appear in our sequence (n oligomer of a length 3). Also the Markov property, that the future depends on the present state only may not always hold, it could be that it depends on more than 1 state before the present state. Markov chains also don’t account mutations, and we also don’t know for sure if we have time homogeneity.