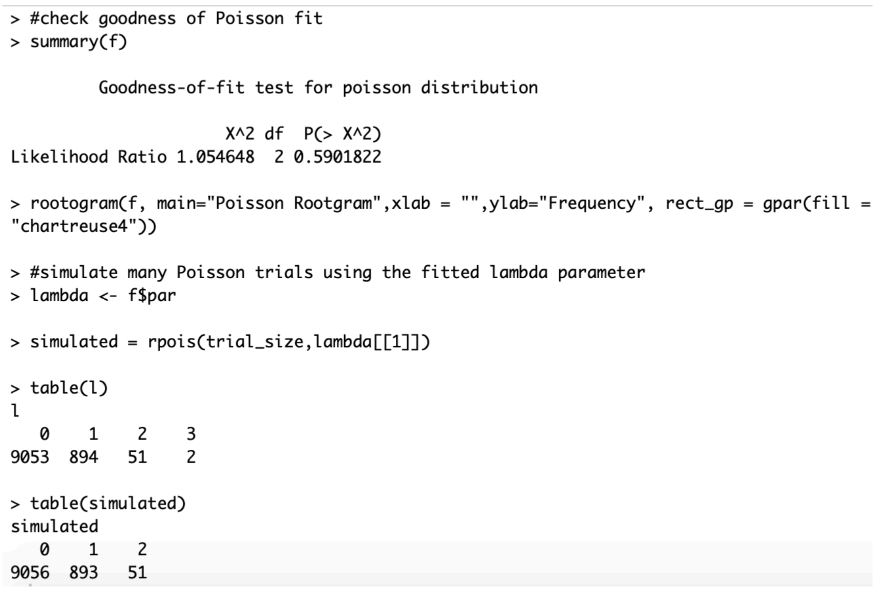
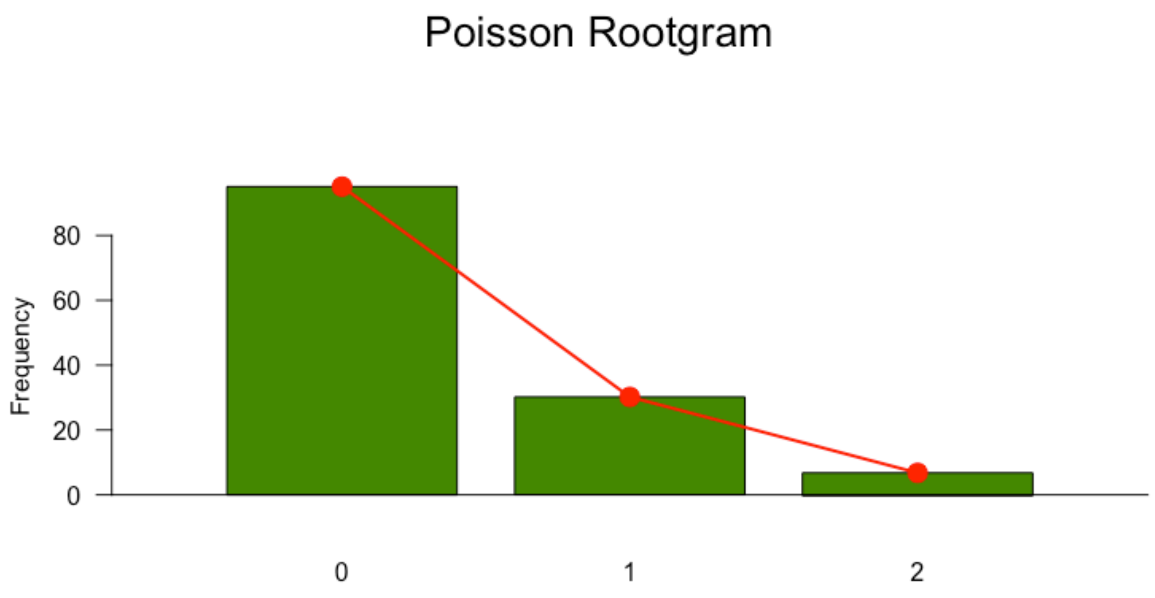
2.1

In this question, n is large and p is small; therefore, the distribution should be Poisson.

From the goodness-of-fit summary, we accept the null hypothesis that the underlying process is Poisson. The table comparison and rootgram plot can also confirm this.



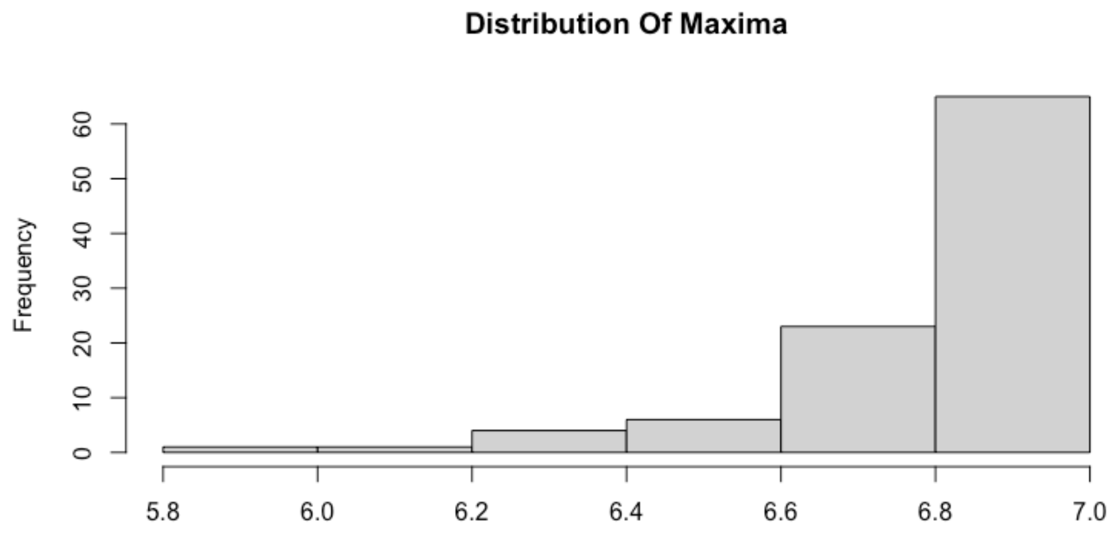
```
> #check goodness of Poisson fit
> summary(f)

         Goodness-of-fit test for poisson distribution

                      X^2 df  P(> X^2)
Likelihood Ratio 1.054648  2 0.5901822

> rootogram(f, main="Poisson Rootgram",xlab = "",ylab="Frequency", rect_gp = gpar(fill =
"chartreuse4"))

> #simulate many Poisson trials using the fitted lambda parameter
> lambda <- f$par

> simulated = rpois(trial_size,lambda[[1]])

> table(l)
l
   0    1    2    3
9053  894   51    2

> table(simulated)
simulated
   0    1    2
9056  893   51
```



R Code:

```
#2.1
#generate data
sequence_length <- 1000
mutation_rate <- 10^-4
trial_size <- 10000
l <- replicate(trial_size,{sum(rbinom(sequence_length,1,mutation_rate))})
#fit data
library("vcd")
f <- goodfit( l, "poisson")
#check goodness of Poisson fit
summary(f)
rootogram(f, main="Poisson Rootgram",xlab = "",ylab="Frequency", rect_gp = gpar(fill = "chartreuse4"))

#simulate many Poisson trials using the fitted lambda parameter
lambda <- f$par
simulated = rpois(trial_size,lambda[[1]])
table(l)
table(simulated)
```

2.2



Based on the plot above, it is evident that the maximum likelihood estimation for $\hat{\theta}$ is 7 for the maximum of 25 independent identically distribution uniform random variable.

The theoretical justification is:

Let X_i be an independent identically distributed uniform random variable, $Unif(a,b), a < b$, and $Y_n = \max(X_1, X_2, \ldots, X_n)$

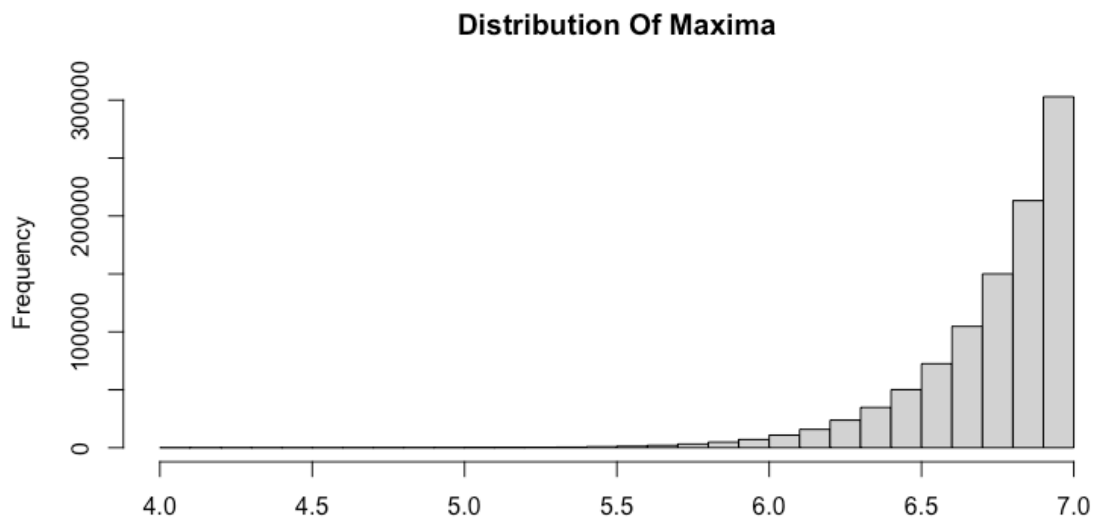
$$p(Y_n \leq x) = p(X_1 \leq x \,\&\, X_2 \leq x \,\&\, \ldots \,\&\, X_n \leq x) = (\frac{x-a}{b-a})^n$$

Let δ be an infinitesimally small number:

$$\lim_{n\to\infty} p(Y_n \leq b-\delta) = (\frac{b-a-\delta}{b-a})^n = 0$$

Therefore, as n increases, the maximum converges to $b = 7$

Let's increase the number of trials B from 100 to 10000, and plot the distribution of maxima again:



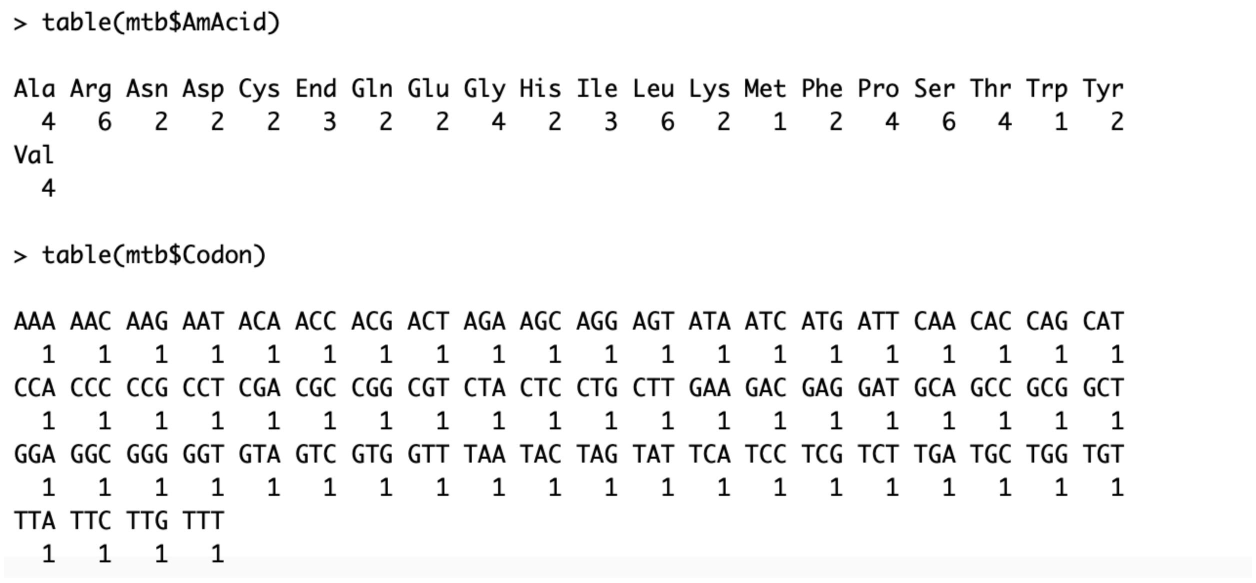
R Code:

```
#2.2
#generate data
generator <- function(n=25,min=0,max=7){
  return(max(runif(n,0,7)))
}
B = 100
l <- replicate(B,generator())
#plot data
hist(l,xlab="",main="Distribution Of Maxima")
```

2.3

a.

The fact that 20 amino acids have redundant expressions due to 64 codon spellings is verified below.



```
> table(mtb$AmAcid)

Ala Arg Asn Asp Cys End Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser Thr Trp Tyr 
  4   6   2   2   2   3   2   2   4   2   3   6   2   1   2   4   6   4   1   2 
Val 
  4 

> table(mtb$Codon)

AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA ATC ATG ATT CAA CAC CAG CAT 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
CCA CCC CCG CCT CGA CGC CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC GCG GCT 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT TCA TCC TCG TCT TGA TGC TGG TGT 
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
TTA TTC TTG TTT 
  1   1   1   1 
```

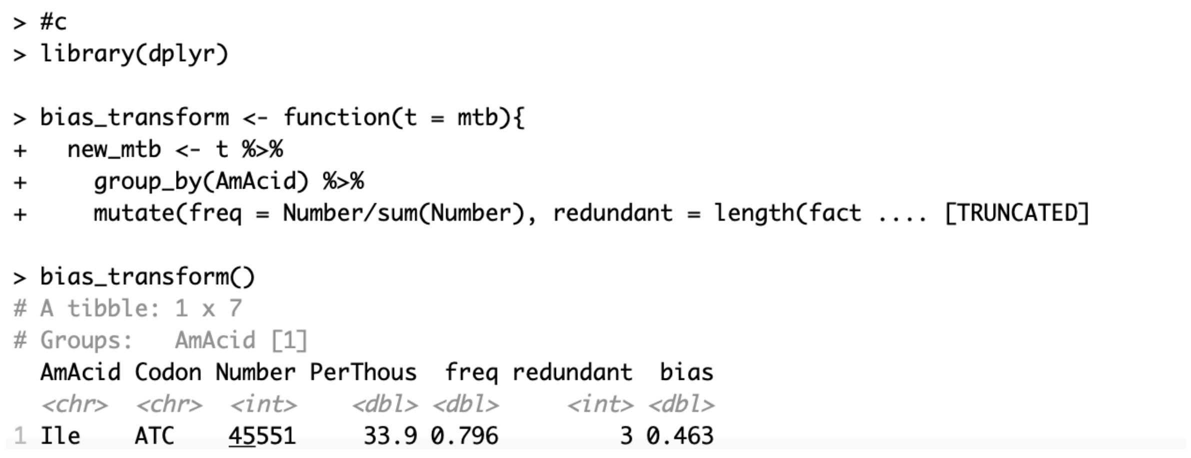
b.

The “PerThous” variable refers to the frequency that a codon would appear every thousand codons.

It can be computed from the command “(mtb$Number/sum(mtb$Number))*1000”

c.

the strongest bias belongs to the isoleucine amino acid. There are three codon spellings for isoleucine and the greatest bias for ATC is 46.3%.



```
> #c
> library(dplyr)

> bias_transform <- function(t = mtb){
+   new_mtb <- t %>%
+     group_by(AmAcid) %>%
+     mutate(freq = Number/sum(Number), redundant = length(fact .... [TRUNCATED]

> bias_transform()
# A tibble: 1 x 7
# Groups:   AmAcid [1]
  AmAcid Codon Number PerThous  freq redundant  bias
  <chr>  <chr>  <int>    <dbl> <dbl>     <int> <dbl>
1 Ile    ATC    45551     33.9 0.796         3 0.463
```

R Code:

```
#2.3
#a
mtb = read.table("~/Desktop/M_tuberculosis.txt",header=TRUE)
table(mtb$AmAcid)
table(mtb$Codon)

#b
(mtb$Number/sum(mtb$Number))*1000

#c
library(dplyr)

bias_transform <- function(t = mtb){
  new_mtb <- t %>%
    group_by(AmAcid) %>%
    mutate(freq = Number/sum(Number), redundant = length(factor(Codon))) %>%
    mutate(bias = abs(freq-(1/redundant)))

  return(new_mtb[which.max(new_mtb$bias),])}

bias_transform()
```

2.4

Question 2.4 was completed with help from a tutorial on “Biostring” posted by the Stanford University: https://web.stanford.edu/class/bios221/labs/biostrings/lab_1_biostrings.html

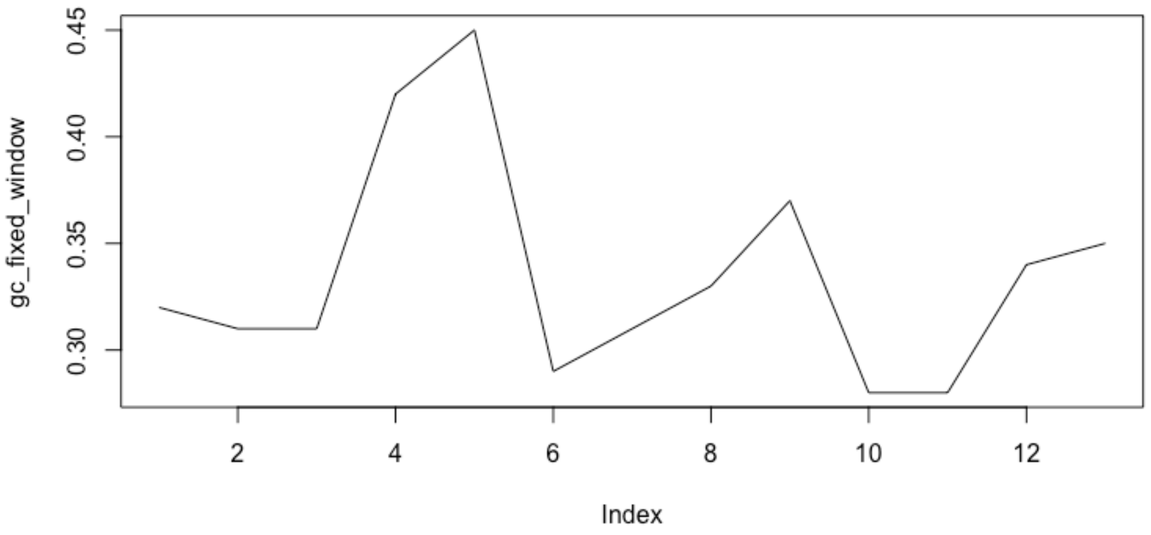
a.

To see the complete sequence, use the “as.character(staph[i])” expression.

```
> #a
> staph = readDNAStringSet("~/Desktop/staphsequence.ffn.txt", "fasta")

> staph[1:3]
DNAStringSet object of length 3:
    width seq                                               names
[1]  1362 ATGTCGGAAAAAGAAATTTGGGAAA...GAAAAAGAAATAAGAAATGTATAA lcl|NC_002952.2_c...
[2]  1134 ATGATGGAATTCACTATTAAAAGAG...ATTTTACCAATCAGAACTTACTAA lcl|NC_002952.2_c...
[3]   246 GTGATTATTTTGGTTCAAGAAGTTG...ATCATTCATCAAGGTGAACAATGA lcl|NC_002952.2_c...
```

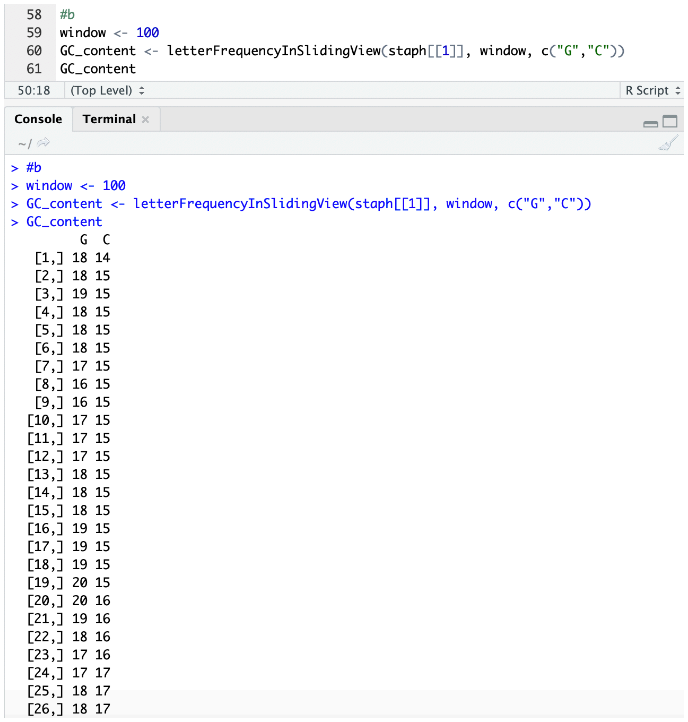
b.

Herein we use the built-in function “alphabetFrequency” from the package “Biostring” for fixed window analysis



c.

Herein we use the built-in function "letterFrequencyInSlidingWindow" from the package "Biostring" for sliding window analysis

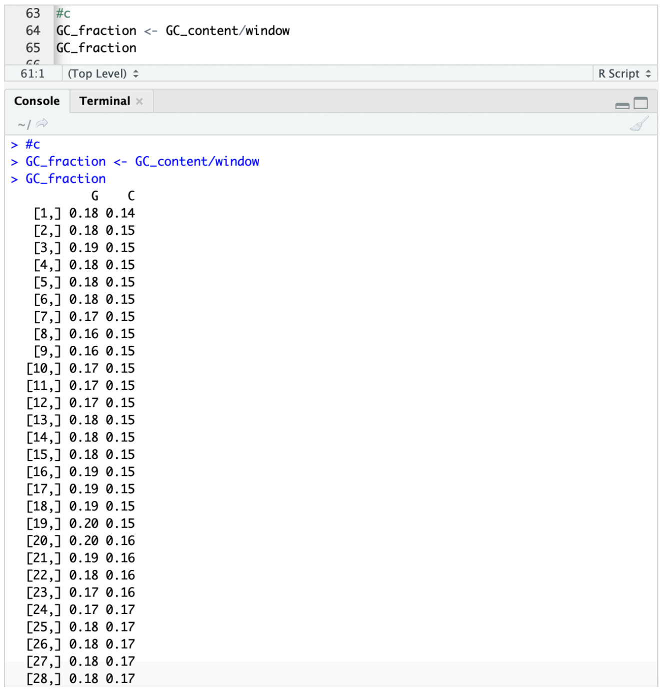


```
#b
window <- 100
GC_content <- letterFrequencyInSlidingView(staph[[1]], window, c("G","C"))
GC_content
```

50:18 (Top Level) R Script

Console Terminal

```
> #b
> window <- 100
> GC_content <- letterFrequencyInSlidingView(staph[[1]], window, c("G","C"))
> GC_content
       G  C
 [1,] 18 14
 [2,] 18 15
 [3,] 19 15
 [4,] 18 15
 [5,] 18 15
 [6,] 18 15
 [7,] 17 15
 [8,] 16 15
 [9,] 16 15
[10,] 17 15
[11,] 17 15
[12,] 17 15
[13,] 18 15
[14,] 18 15
[15,] 18 15
[16,] 19 15
[17,] 19 15
[18,] 19 15
[19,] 20 15
[20,] 20 16
[21,] 19 16
[22,] 18 16
[23,] 17 16
[24,] 17 17
[25,] 18 17
[26,] 18 17
```



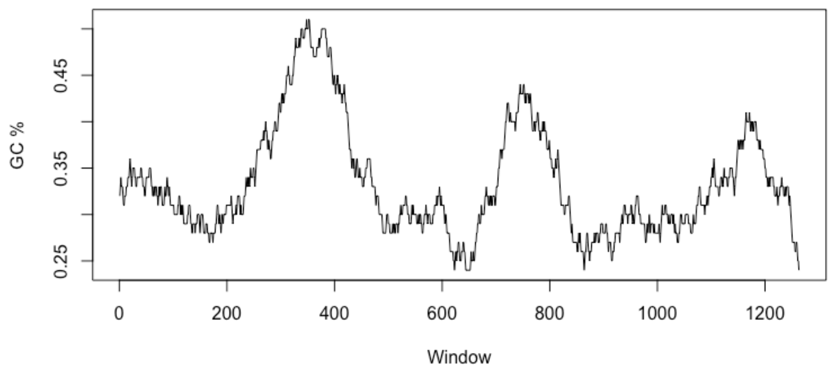
```
#c
GC_fraction <- GC_content/window
GC_fraction
```

61:1 (Top Level) R Script

Console Terminal

```
> #c
> GC_fraction <- GC_content/window
> GC_fraction
         G    C
 [1,] 0.18 0.14
 [2,] 0.18 0.15
 [3,] 0.19 0.15
 [4,] 0.18 0.15
 [5,] 0.18 0.15
 [6,] 0.18 0.15
 [7,] 0.17 0.15
 [8,] 0.16 0.15
 [9,] 0.16 0.15
[10,] 0.17 0.15
[11,] 0.17 0.15
[12,] 0.17 0.15
[13,] 0.18 0.15
[14,] 0.18 0.15
[15,] 0.18 0.15
[16,] 0.19 0.15
[17,] 0.19 0.15
[18,] 0.19 0.15
[19,] 0.20 0.15
[20,] 0.20 0.16
[21,] 0.19 0.16
[22,] 0.18 0.16
[23,] 0.17 0.16
[24,] 0.17 0.17
[25,] 0.18 0.17
[26,] 0.18 0.17
[27,] 0.18 0.17
[28,] 0.18 0.17
```

d.
We could plot the GC fraction along the window sequence.
Here we can see the plot from part (d) roughly follows that of (b).



R Code:

```
#2.4
library("Biostrings")

#a
staph = readDNAStringSet("~/Desktop/staphsequence.ffn.txt", "fasta")
staph[1:3]

#b
library(Biostrings)
staph <- readDNAStringSet("~/Desktop/staphsequence.ffn.txt", "fasta")

window <- 100
l <- length(staph[[1]])
start <- (c(1:as.integer(l/window))-1)*window
end <- start + window

view <- Views(staph[[1]],start=start,end=end)
```

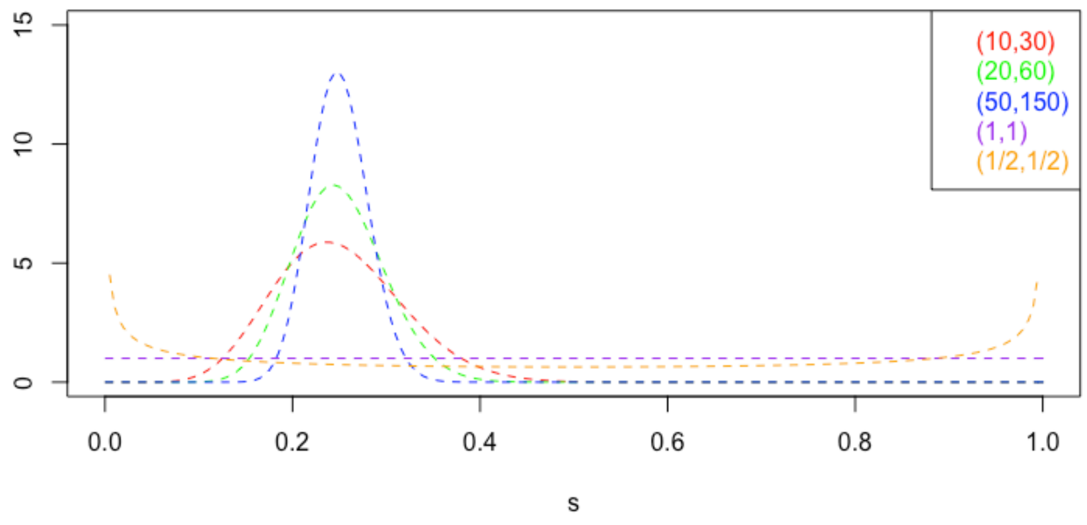
```
gc_fixed_window <- rowSums(alphabetFrequency(view)[, c(2,3)]/window)
plot(gc_fixed_window, type = 'l')
#c
window <- 100
GC_content <- letterFrequencyInSlidingView(staph[[1]], window, c("G","C"))
GC_content
GC_fraction <- GC_content/window
GC_fraction

#d
GC_roll <- rowSums(GC_fraction)
plot(GC_roll, type = 'l',ylab="GC %",xlab="Window")
```

2.5

$B(1,1)$ is flat, hence the name "uniform" distribution.

$B(\frac{1}{2},\frac{1}{2})$ is flat at its central region then curves up at its tails.

Using formula $E(X)=\frac{\alpha}{\alpha+\beta}$, we see that $B(\frac{1}{2},\frac{1}{2})$ and $B(1,1)$ have the same mean at 0.5 while the rest of the Bs have the same mean at 0.25.



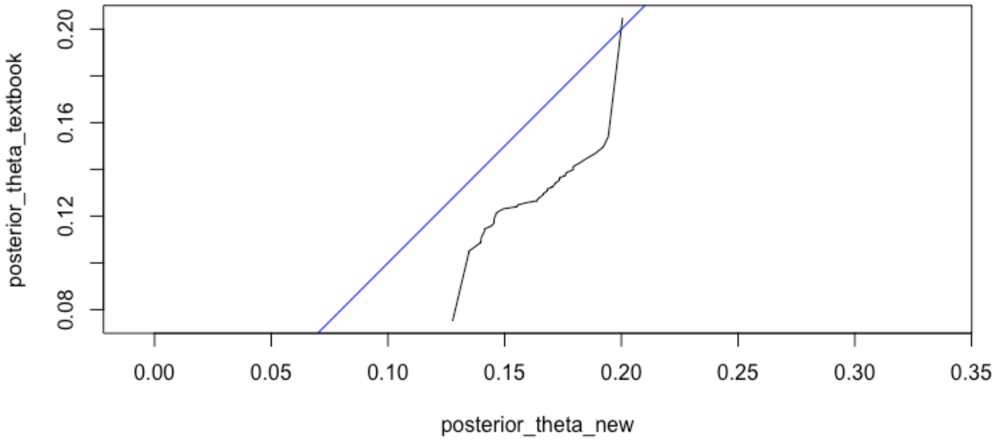
R Code:

```
s <- seq(0,1,by=0.005)
d_10_30 <- dbeta(s, 10, 30)
d_20_60 <- dbeta(s, 20, 60)
d_50_150 <- dbeta(s, 50, 150)
d_1_1 <- dbeta(s, 1, 1)
d_h_h <- dbeta(s, 1/2, 1/2)

plot(s,d_10_30,type="l",lty=2,ylab="",ylim=c(0,15),col="red")
lines(s,d_20_60,col="green",lty=2)
lines(s,d_50_150,col="blue",lty=2)
lines(s,d_1_1, col="purple",lty=2)
lines(s,d_h_h, col="orange",lty=2)

legend( "topright",
c("(10,30)","(20,60)","(50,150)","(1,1)","(1/2,1/2)"),text.col=c("red","green","blue","purple","o
range"))
```

2.6

The prior distribution for the textbook example is $B(50,350)$ and its posterior distribution given $n = 300, Y = 40$ is $B(90,610)$.

I chose the prior distribution to be $B(20,50)$ and plotted its posterior against that from the textbook example above in a QQ plot.



As expected, they are not matched because the theoretical posterior distribution should be $B(60,310)$, quite different from $B(90,610)$.

R Code:

```
#2.6
#the posterior distribution from textbook where alpha = 50, beta = 350
posterior_theta_textbook = rbeta(n = 1e6, 90, 610)
#my own posterior distribution generated from alpha = 20, beta= 50
rtheta = rbeta(100000, 20, 50)
y = vapply(rtheta, function(th) {rbinom(1, prob = th, size = 300)}, numeric(1))
posterior_theta_new = rtheta[ y == 40 ]
qqplot(posterior_theta_new, posterior_theta_textbook, type = "l", asp = 1)
abline(a = 0, b = 1, col = "blue")
```