FINAL PROJECT

TOPIC 1

Read and clean data (fasta format)

```
#read data
a = read.table("67.txt")

#get the lines with sequence only
linesWithSeq = grep(pattern="^>", x = a[,1], invert = TRUE)

#strsplit each sequence to handle them easier
data = strsplit(a[linesWithSeq,],"")
```

QUESTION 1 :

Find all instances of the given motif in the sequences, for which the Hamming distance between the substring and the motif is less than 2.

The motif has 6 positions.

Get usefuldata/all sequences with 6 or more positions.

```
usefuldata = list()

counter = 1

for (i in 1:length(data)){

  if (length(data[[i]]) >= 6){

    usefuldata[[counter]] = data[[i]]
    counter = counter + 1
  }
}

length(data)-length(usefuldata) # only 2 sequences with less than 6 elements
```

```
## [1] 2
```

From each sequence, we get all the substrings with length 6, that have haming distance < 2 from the motif.

```
substrings = list()
index = list()
len = list ()
idx = 1

#function to get all the substrings with length 6 that have haming distance < 2 from the motif.
Hammingdistance = function(x){

  # Given motif
  motif = c("C","G","T","C","A","C")

  # for each sequence substrings with length 6 will be in total:
  #length(sequence) - length(motif) + 1

  for (i in 1:(length(x) - length(motif) + 1)){

    # x[i:(i+5)] is the current substring

    #hamming distance between the motif  and the substring
    dif = sum(motif != x[i:(i+5)])

    #if hamming distance < 2:
    if (dif == 0 | dif == 1){

      # get the substring
      substrings[[idx]] <<- x[i:(i+5)]

      # get substrings starting position
      index[[idx]] <<- i

      # get the length of the sequence  that this substring belongs to
      len[[idx]] <<- length(x)

      idx <<- idx + 1
    }
  }

}

# apply that function to usefuldata to get substrings,index and len
invisible(lapply(usefuldata,Hammingdistance))

# variable substrings contains the substrings for which the hamming distance between them
# and  the motif is less than 2
```

QUESTION 2 : Construct the PWM using these substrings

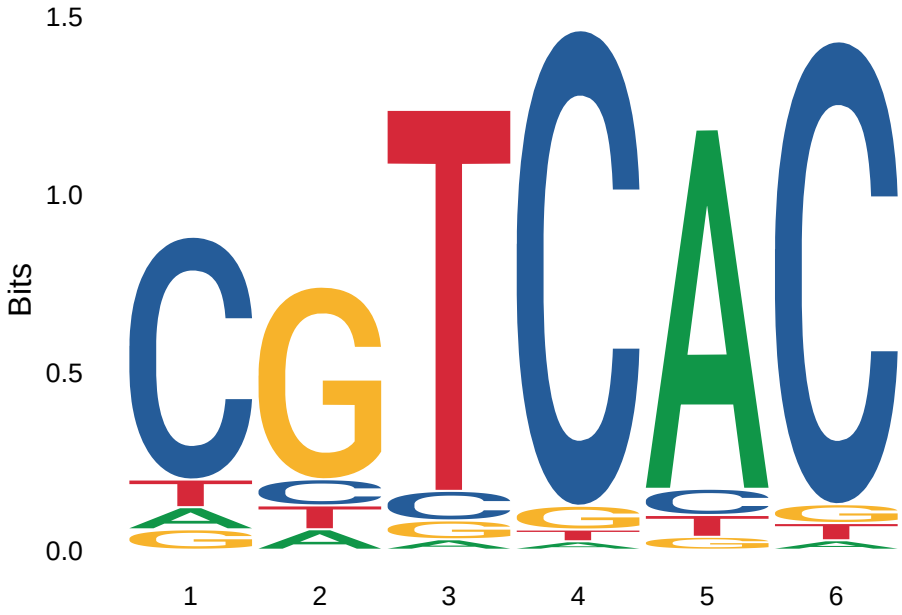
```
mtrx = t(matrix(unlist(substrings) , nrow = 6))

pfm = matrix(0, nrow=4, ncol=6)

rownames(pfm) = c("A","C","G","T")

# count the number of appearances of a letter in each position
pfm = apply(mtrx, 2, function(x){table(factor(x, levels= c("A","C","G","T")))})

# make count to frequency
ppm = pfm/as.vector(apply(pfm, 2, sum))

#  divide by 1/4 and take the log2
pwm = log2(ppm/0.25)

pwm
```

```
##        [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## A -1.820648 -1.685316 -3.400316 -3.948109  1.774115 -3.753293
## C  1.633591 -1.329738 -1.895487  1.872531 -2.002724  1.864719
## G -1.940885  1.550296 -2.541789 -2.450730 -3.025468 -2.753293
## T -1.499781 -1.464132  1.792762 -3.465168 -2.269675 -2.882972
```

QUESTION 3 : Construct the logo for the PPM

In each position in the logo,the biggest so the most frequent letter, is the one that corresponds in the given motif CGTCAC

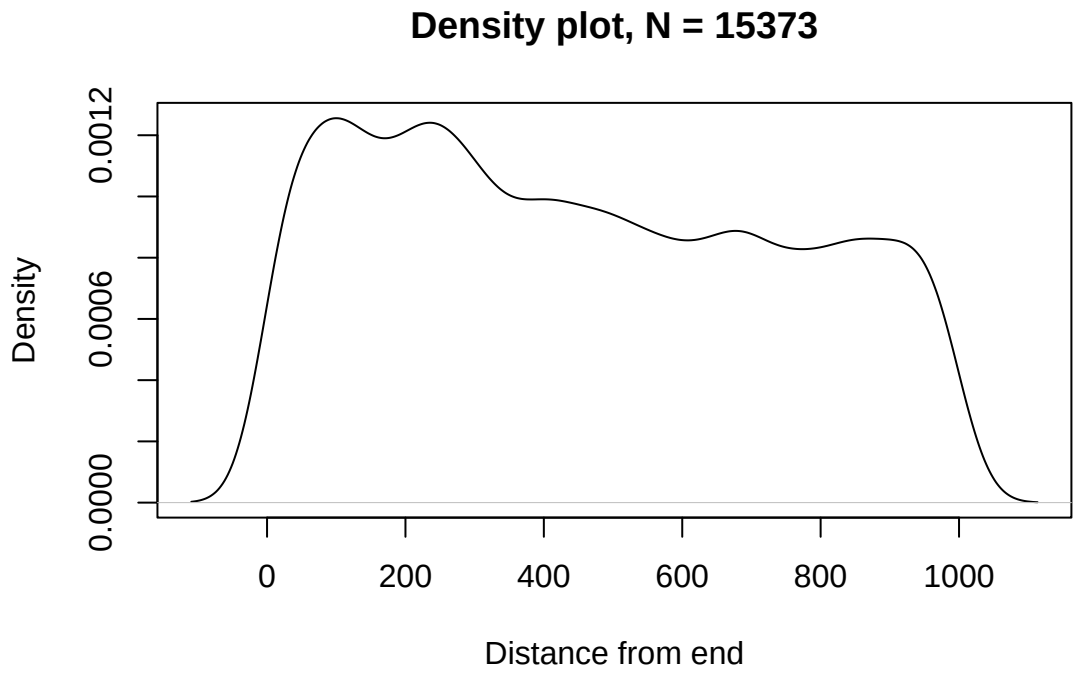
```
require(ggplot2)
require(ggseqlogo)
ggseqlogo(ppm,method='bits')
```



QUESTION 4 : How far is the starting position of each substring from the end of the sequence?

Construct their density plot

```
position_from_theend = unlist(len) - unlist(index)

plot(density(position_from_theend),main = "Density plot, N = 15373",xlab ="Distance from end")
```



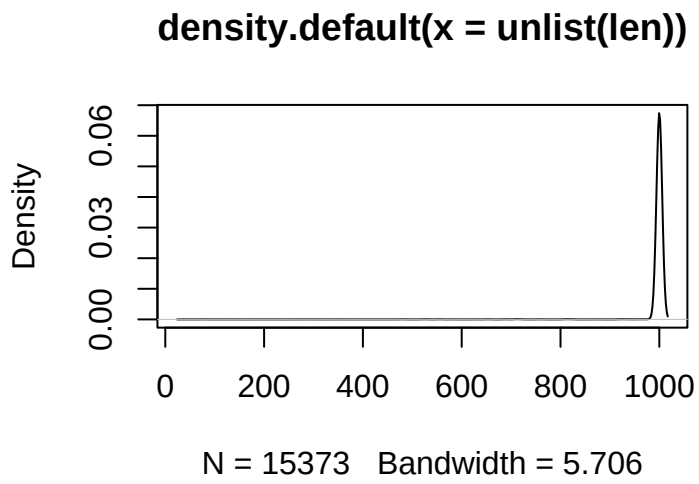

```
length(substrings)
```

```
## [1] 15373
```

What do you observe?

First lets plot the distribution for the length of the sequences

```
plot(density(unlist(len)))
```




```
# percentage of sequences with less than 1000 positions
length( which(unlist(len)<1000) )/length(unlist(len))
```

```
## [1] 0.01183894
```

Only 1.1% of the sequences have length less than 1000.

That means that, if distances from the end were random, the distribution of the distances would follow the uniform distribution in range [5,1000]

Instead we see a right skew in the distribution of the distances,meaning that there are more substrings (with hamming distance < 2 from the motif), with starting positions closer to the end of the sequences.

Topic 2

QUESTIONS 1,2 : Clean and Log the data

```
# read lines of our dataset
dataset = readLines("GDS3713.soft")

# remove all lines that start with ^, #, !
cleanlines = dataset[grep('^[/!#/^]', dataset, invert = TRUE)]

# make new file with our clean dataset
write(cleanlines,"GDS3713.soft.clean",ncolumns = length(cleanlines[1]), sep="\t")

data=read.table("GDS3713.soft.clean",header = TRUE)

# remove the first 2 columns that are not expression values
data=data[,-c(1,2)]

# log10 the data
data=log10(data)
```

QUESTION 3 : which samples are smokers and which are control?

According to the experiment info, first 40 samples are control and the next 39 are smokers.

QUESTION 4 : Find all genes with expression value statistically higher in smokers than control. Use t.test. The significance threshold should be 0.05 after FDR correction

The alternative hypothesis H1 is : smokers > control

Get the pvalues

```
# apply on every row/gene of the data
getpval<- function(row){
  # first 40 samples are control
  control_i=row[1:40]
  # last 39 are smokers
  smokers_i=row[41:79]

  # get pvalue for H1: smokers > control  (t test)
  pvalue = t.test(smokers_i,control_i,alternative = 'greater')$p.value
  return(pvalue)
}

# get the pvalues
pvalues = apply(data, 1, getpval)

# apply FDR correction
pvalues.fdr=p.adjust(pvalues,method = "fdr")
```

Genes with adjusted pvalues < 0.05 have value statistically higher in smokers than non-smokers.

Get the genes with adjusted pvalues <0.05

```
genes_higher_smokers=which(pvalues.fdr<0.05)
length(genes_higher_smokers)
```

```
## [1] 1368
```

TOPIC 3 : Follow the ABC methodology with the Euclidean Distance and use the simulated datasets to infer the growth rate for the observation.

Function to read ms files

```
# Read ms files
read.ms.output <- function( file.ms.output=NA ) {

  txt=NA

  if( !is.na(file.ms.output) ) txt <- scan(file=file.ms.output,
                                           what="character", sep="\n", quiet=TRUE)
  if( is.na(txt[1]) ){
    print("Usage: read.ms.output(txt), or read.ms.output(file=filename)")
    return()
  }
  nsam <- as.integer( strsplit(txt[1], split=" ")[[1]][2] )
  ndraws <- as.integer( strsplit( txt[1], split=" ")[[1]][3] )

  h <- numeric()
  result <- list()
  gamlist <- list()
  positions <- list()

  marker <- grep("prob",txt)
  probs <- sapply(strsplit(txt[marker], split=":"), function(vec) as.numeric(vec[2]))
  marker <- grep("time",txt)
  times <- sapply(strsplit(txt[marker], split="\t"), function(vec){ as.numeric(vec[2:3])} )


  ## THE OUTPUT TEXT FOR EACH DRAW SHOULD CONTAIN THE WORD "segsites"
  marker <- grep("segsites", txt)

  if( length(marker) != ndraws){
    stop( paste("length: ", length(marker), " ndraws: ", ndraws) )
    stopifnot(length(marker) == ndraws)
  }

  ## GET NUMBERS OF SEGREGATING SITES IN EACH DRAW
  segsites <- sapply(strsplit(txt[marker], split=" "), function(vec) as.integer(vec[2]) )
  for(draw in seq(along=marker)) {
    if(!(draw %% 100)) cat(draw, " ")
    if(segsites[draw] > 0) {
      tpos <- strsplit(txt[marker[draw]+1], split=" ")
```

```
      positions[[draw]] <- as.numeric( tpos[[1]][ 2:(segsites[draw]+1) ] )
      haplotypes <- txt[(marker[draw] + 2):(marker[draw] + 2 + nsam - 1)]
      haplotypes <- strsplit(haplotypes, split="")
      h <- sapply(haplotypes, function(el) c(as.integer(el)))
      ## IF THERE'S 1 SEGREGATING SITE, THIS WON'T BE A MATRIX
      if(segsites[draw] == 1) h <- as.matrix(h)
      ## OTHERWISE, IT NEEDS TO BE TRANSPOSED
      else h <- t(h)
    }
    else {
      h <- matrix(nrow=nsam, ncol=0)
      positions[[draw]]<- NA
    }
    gamlist[[draw]] <- h
    stopifnot(all(dim(h) == c(nsam, segsites[draw])))
  }
  cat("\n")
  list(segsites=segsites, gametes=gamlist, probs=probs, times=t(times), positions=positions, nsam=nsam,
}
```

Get simulation data

```
#download.file("http://139.91.162.101//teaching/project2022/R/ms/ms.sim.out", "ms.sim.out")
invisible(simdata<- read.ms.output("ms.sim.out"))
```

```
## 100  200  300  400  500  600  700  800  900  1000  1100  1200  1300  1400  1500  1600  1700  1800  19
```

CASE 1 : Use as summary statistics the i) average number of pairwise differences between sequences, ii) number of SNPs.

```
reps=length(simdata$gametes)
#matrix for the summary statistics
all.sim.stats = matrix(0, nrow=reps, ncol=2)
```

Calculate the number of SNPs (Single-nucleotide polymorphism)

```
# function to calculate number of SNPs
snp=function(data){
  snps=0
  for (i in 1:(ncol(data))){
    #an uparxei 1 sthn sthlh tote einai polumorfikh
    if(length(which(data[,i]==1))>0){
      snps=snps+1
    }
  }
  return(snps)
}
#save SNPs as the first summary statistic
all.sim.stats[,1]=sapply(simdata$gametes,snp)
```

Calculate the average number of pairwise differences between sequences

```
pairdif = function(data){
  dif = 0
  for (i in 1:(nrow(data)-1)){
    for(j in (i+1):(nrow(data))){
      dif = dif + sum(data[i,] != data[j,])
    }
  }
  return(2*dif/(nrow(data)*(nrow(data)-1)))
}

#save pairwise differences as the second summary statistic
all.sim.stats[,2]=as.numeric(lapply(simdata$gametes, pairdif))
```

Get the observation and its statistics

```
#download.file("http://139.91.162.101//teaching/project2022/R/ms/ms.obs.out","ms.obs.out")

#read the observation data:
ms <- read.ms.output("ms.obs.out")
```

```
data = ms$gametes[[1]]

#statistics of the observation  (snps,pairdif)
stats = c(snp(ms$gametes[[1]]), pairdif(ms$gametes[[1]]))
```

We want the euclidean distance between the statistics of the simulations and the observation

```
# euclidean distance function
eucl = function(m, obs){
  difs = apply(m, 1, function(x){ sqrt(sum((obs - x)^2)) })
  return(difs)
}

# euclidean distances for the statics of the observation and the simulations
difs = eucl(all.sim.stats, stats)
```

Finally we get the closest 500 simulations to observation (i.e. the 500 simulations with the least euclidean distance)

```
# get closest 500 simulations to observation by sorting the euclidean distances
accepted.dif.indexes  = which(difs <= sort(difs)[500] )
```

Read the growth rate values of the simulations (params)

```
#download.file("http://139.91.162.101//teaching/project2022/R/ms/params.txt","params.txt")
params=read.table("params.txt")
```

The prior values are all the growth rate values (params).

The posterior values, are the growth rate values for the closest 500 simulations

```
posterior=params[accepted.dif.indexes,]

#prior density
d.prior=density(params[,1])

#posterior density
d.posterior=density(posterior)
```

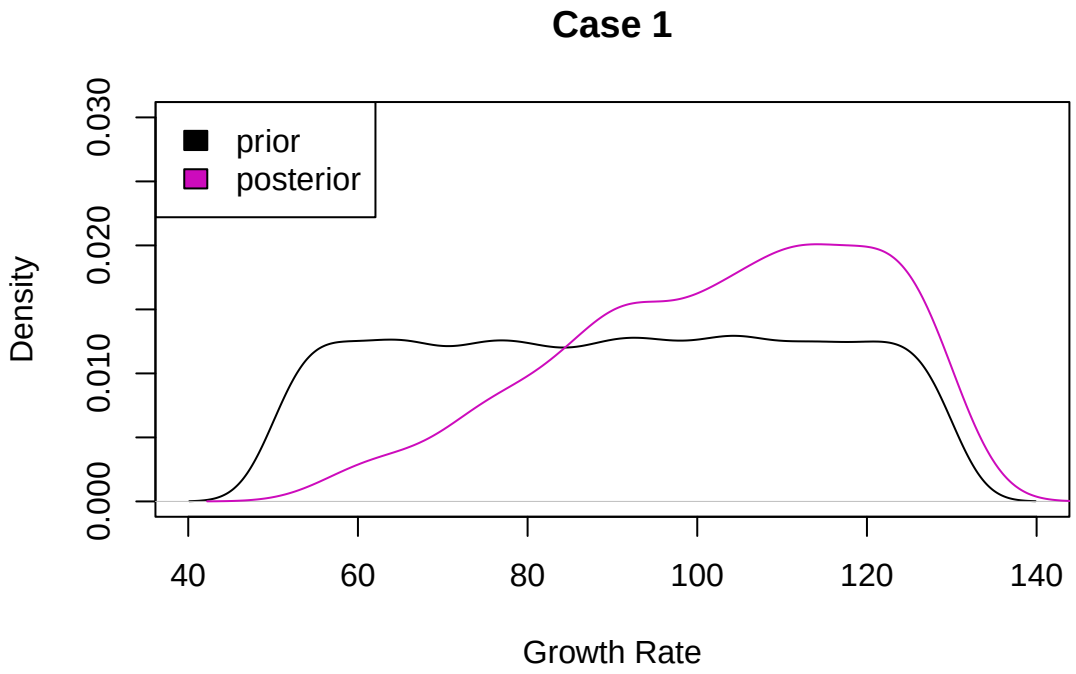
Now ,we can infer the growth rate of the observation, as the mean of the posterior values.

```
# our estimate for the growth rate G
case1_infer=mean(posterior)
case1_infer
```

```
## [1] 102.7754
```

Density plot for the prior and the posterior

```
plot(d.prior, col='1',ylim=c(0,0.03),main="Case 1", xlab="Growth Rate")

points(d.posterior, col='6', type='l')

legend('topleft',legend=c("prior",'posterior'),fill=c(1,6))
```



CASE 2 : Use one more statistic, the Tajima's D.

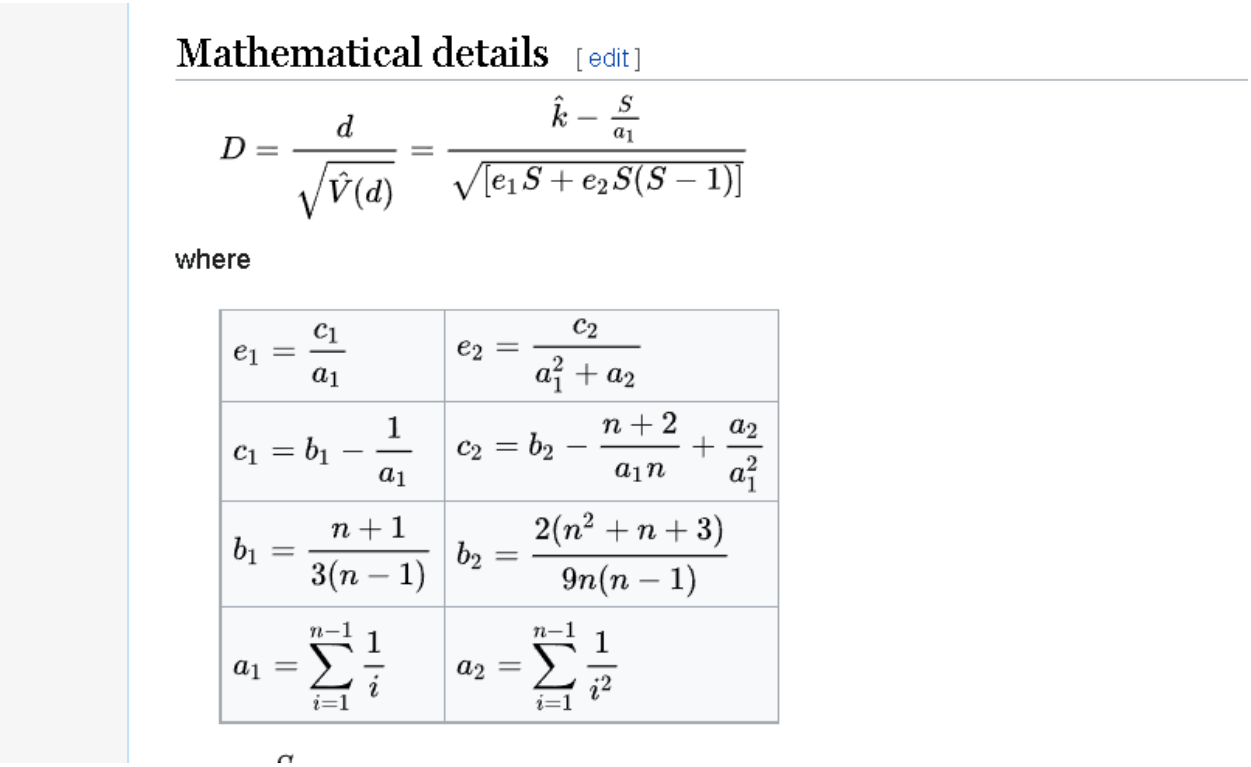
Summary statistics

```
all.sim.stats2 = matrix(0, nrow=reps, ncol=3)

# first 2 statistics are the same as Case 1
all.sim.stats2[,1:2]=all.sim.stats
```

For Tajimas D:

$$\hat{k} = \texttt{Average number of pairwise differences}$$

$$S = \texttt{Number of SNPs}$$



Mathematical details [edit]

$$D = \frac{d}{\sqrt{\hat{V}(d)}} = \frac{\hat{k} - \frac{S}{a_1}}{\sqrt{[e_1 S + e_2 S(S-1)]}}$$

where

$e_1 = \frac{c_1}{a_1}$	$e_2 = \frac{c_2}{a_1^2 + a_2}$
$c_1 = b_1 - \frac{1}{a_1}$	$c_2 = b_2 - \frac{n+2}{a_1 n} + \frac{a_2}{a_1^2}$
$b_1 = \frac{n+1}{3(n-1)}$	$b_2 = \frac{2(n^2+n+3)}{9n(n-1)}$
$a_1 = \sum_{i=1}^{n-1} \frac{1}{i}$	$a_2 = \sum_{i=1}^{n-1} \frac{1}{i^2}$

Figure 1: Tajimas D,Wikipedia

```
# mapply this on the simulation data to get Tajimas D
tajimasd=function (x,khat,S){
  n=nrow(x)
  tmp <- 1:(n - 1)
  a1 <- sum(1/tmp)
  a2 <- sum(1/tmp^2)
  b1 <- (n + 1)/(3 * (n - 1))
  b2 <- 2 * (n^2 + n + 3)/(9 * n * (n - 1))
  c1 <- b1 - 1/a1
  c2 <- b2 - (n + 2)/(a1 * n) + a2/a1^2
  e1 <- c1/a1
  e2 <- c2/(a1^2 + a2)
  D <- (khat - S/a1)/sqrt(e1 * S + e2 * S * (S - 1))
  return(D)
}


# save tajimas D as the 3rd statistic
all.sim.stats2[,3]=mapply(tajimasd,simdata$gametes,khat=all.sim.stats2[,2],S=all.sim.stats2[,1])
```

For the observation

```
# stats for the observation
stats2= c(snp(ms$gametes[[1]]), pairdif(ms$gametes[[1]]), tajimasd(ms$gametes[[1]],stats[2],stats[1]))


# euclidean distances for the statics of the observation and the simulations
difs2 = eucl(all.sim.stats2, stats2)

# closest 500 simulations to observation
accepted.dif.indexes2= which(difs2 <= sort(difs2)[500] )
```

Case 2 Posterior

```
# Growth rate values for the closest 500 simulations
posterior2=params[accepted.dif.indexes2,]

# posterior density
d.posterior2=density(posterior2)
```

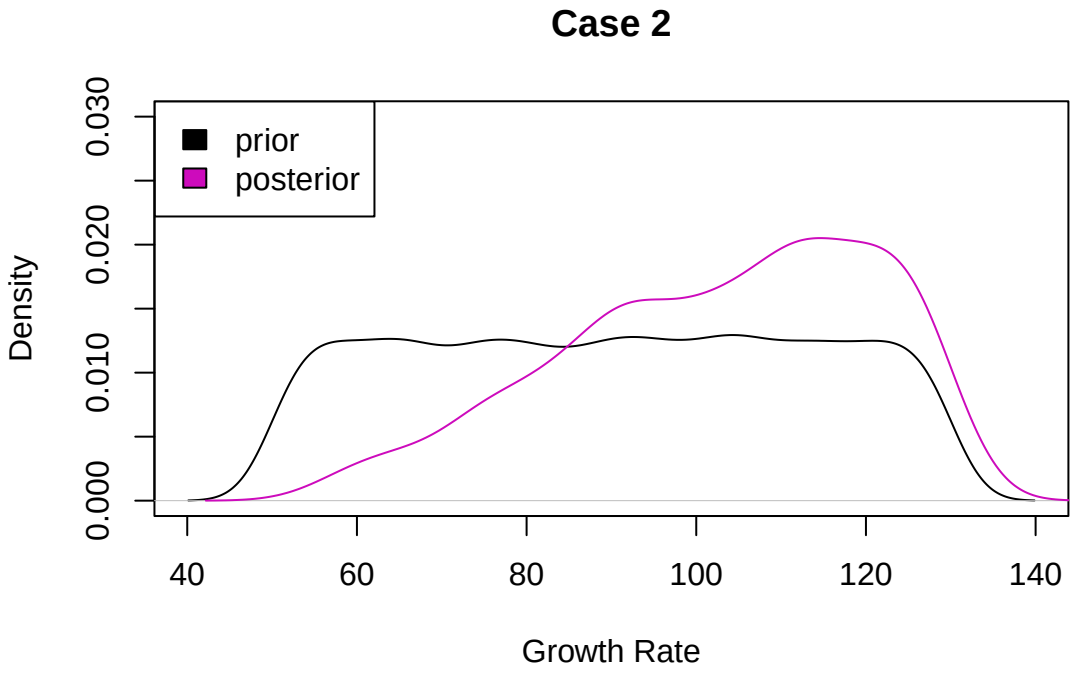
Again we infer the growth rate of the observation, as the mean of the posterior values.

```
# our estimate for the growth rate G

case2_infer=mean(posterior2)
case2_infer
```

```
## [1] 102.818
```

Density Plot for prior and posterior

```
plot(d.prior, col='1',ylim=c(0,0.03),main="Case 2", xlab="Growth Rate")
points(d.posterior2, col='6', type='l')
legend('topleft',legend=c("prior",'posterior'),fill=c(1,6))
```



QUESTION 3 : Which of the inferences (1 or 2) is more accurate if the true value for growth rate is 100?

Case 1 infer value : 102.77

Case 2 infer value : 102.81

The true value for the growth rate parameter of the observation is 100.

In both cases , our infer values are close to the true value, and are very similar,despite using one more summary statistic in case 2.