SLE712 – Bioinformatics and Molecular Biology Techniques

Trimester 1, 2020

Assessment Task Three: Bioinformatics

Lecturer: Dr Mark Ziemann

Student's name: THI LINH CHI TRAN

Student's ID: 220030174

Submit date: 3/06/2020

Part 1: Imported files, data analysis, operations, plots and Github

Reposity link for my assignment: https://github.com/LinhChi1323/SLE712_Assignment_3

Source for part 1: https://github.com/LinhChi1323/SLE712_Assignment_3/tree/master/part1

Question 1

Download file `gene_experession.tsv` from github by `download.file`, read file by `read.csv`, make row name is column one and print first 6 genes by `head()`.

```r
# download the file gene_expression.tsv
download.file("https://github.com/markziemann/SLE712_files/raw/master/bioinfo_asst3_part1_files/gene_exp

# read in the file
df <- read.csv('gene_expression.tsv', sep='\t', stringsAsFactors = FALSE, row.names = 1)

#Try to access a gene by gene name
df['ENSG00000223972', ]
```

```
##                 SRR5150592 SRR5150593
## ENSG00000223972          1          0
```

```r
# show first 6 genes
head(df, 6)
```

```
##                 SRR5150592 SRR5150593
## ENSG00000223972          1          0
## ENSG00000227232          0          1
## ENSG00000278267          0          0
## ENSG00000243485          0          0
## ENSG00000284332          0          0
## ENSG00000237613          0          0
```

Question 2:

Make mean column which is the mean of other columns by `rowMeans`, show first 6 genes by `head()`.

```r
#make mean column
df$mean <- rowMeans(df[, 1:2])

#show first 6 genes
head(df, 6)
```

```
##                  SRR5150592 SRR5150593 mean
## ENSG00000223972          1          0  0.5
## ENSG00000227232          0          1  0.5
## ENSG00000278267          0          0  0.0
## ENSG00000243485          0          0  0.0
## ENSG00000284332          0          0  0.0
## ENSG00000237613          0          0  0.0
```

Question 3

Using `order()`to sort the mean expession from lowest to highest. Take 10 highest genes by `tail()`.

```
# create sorted dataframe by ordered mean column
sorted_df <- df[order(df$mean), ]

# take the name of top 10-highest mean expression genes
top10genes <- row.names(tail(sorted_df, 10))

# look at the result
top10genes
```

```
##  [1] "ENSG00000108821" "ENSG00000198712" "ENSG00000196924" "ENSG00000198786"
##  [5] "ENSG00000198804" "ENSG00000137801" "ENSG00000198886" "ENSG00000075624"
##  [9] "ENSG00000210082" "ENSG00000115414"
```

Question 4

Take mean column after compare this column with 10 (<10). The resutl will be a list of boolean array. Number of genes with mean lower than 10 is sum of this list, which is 43124.

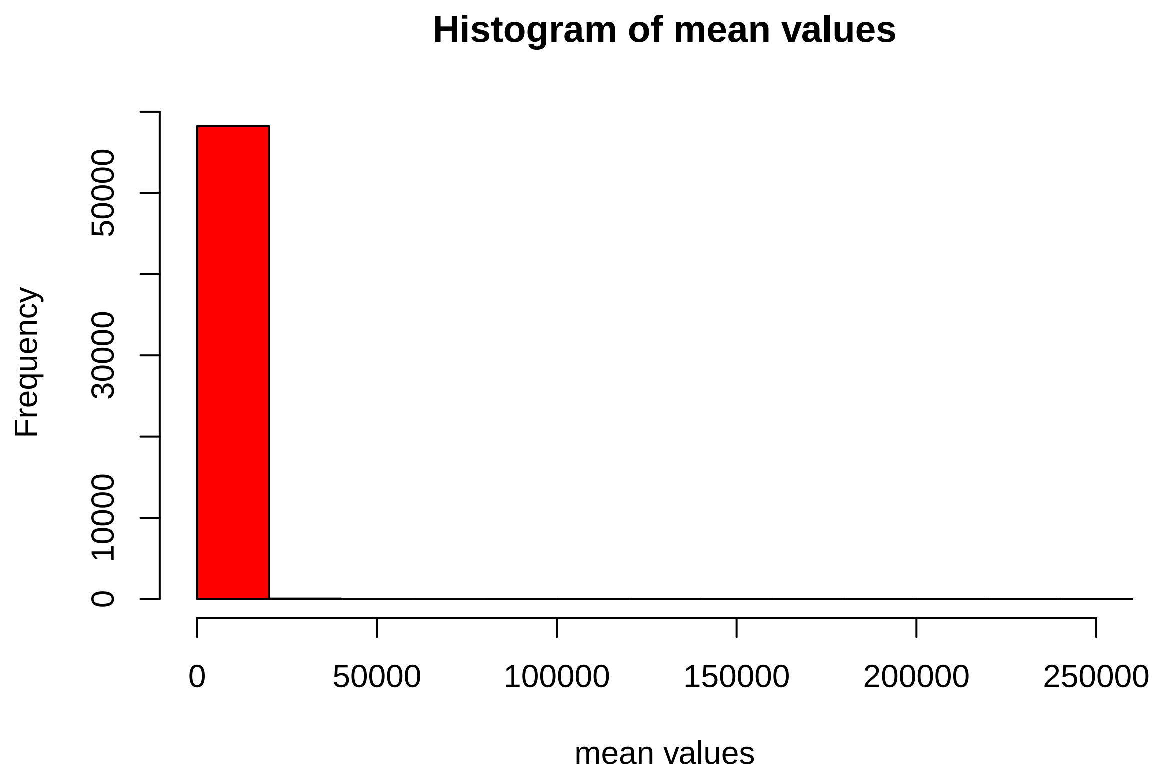
```
number_genes <- sum(df$mean < 10)
cat("The number of genes with a mean < 10: ", number_genes)
```

```
## The number of genes with a mean < 10:  43124
```

Question 5

Using `hist()` function to make a histogram plot and `png` to save it in png format. For my case, I directly show the histogram through running code on Rmarkdown file. In other cases, if you want to save the histogram into a file, `png` and `dev.off` can be used .

```
# png(file = "histogram.png")   # provide histogram file name.
#
# make ahistogram plot of mean values
hist(df$mean, main = "Histogram of mean values",xlab='mean values', col = "red")
```



```
#dev.off()    # save histogram file in png format
```

Question 6

Download growth data from github and load it into a dataframe by `read.csv`. Using command `str` and `head` to confirm that the data has been properly imported. Using `colnames` command to get the column names.

```
# dowload the file growth_data.csv
download.file("https://github.com/markziemann/SLE712_files/raw/master/bioinfo_asst3_part1_files/growth_

# load it into an R object
df <- read.csv('growth_data.csv', header = TRUE, stringsAsFactors = FALSE)

# Print column names of dataframe
colnames(df)
```

```
## [1] "Site"           "TreeID"         "Circumf_2004_cm" "Circumf_2009_cm"
## [5] "Circumf_2014_cm" "Circumf_2019_cm"
```

Question 7

Using `mean` and `sd` to calculate the mean and standard deviation (sd). Mean and sd of tree circumference at the start year (2004) at both sites are 5.077cm and 1.054462 respectively. Mean and sd of tree circumference at the end year (2019) at both sites are 49.912cm and 22.17979 respectively.

```
#Mean and standard deviation at 2004 (start) at both sites
mean_2004 <- mean(df$Circumf_2004_cm)
sd_2004 <- sd(df$Circumf_2004_cm)

cat('Mean at 2004: ', mean_2004)
```

```
## Mean at 2004:  5.077
```

```
cat('\n')
```

```
cat('SD at 2004: ', sd_2004)
```

```
## SD at 2004:  1.054462
```

```
# mean and standard deviation at 2019 (end) at both sites
mean_2019 <- mean(df$Circumf_2019_cm)
sd_2019 <- sd(df$Circumf_2019_cm)

cat('\n')
```

```
cat('Mean at 2019: ', mean_2019)
```

```
## Mean at 2019:  49.912
```

```
cat('\n')
```

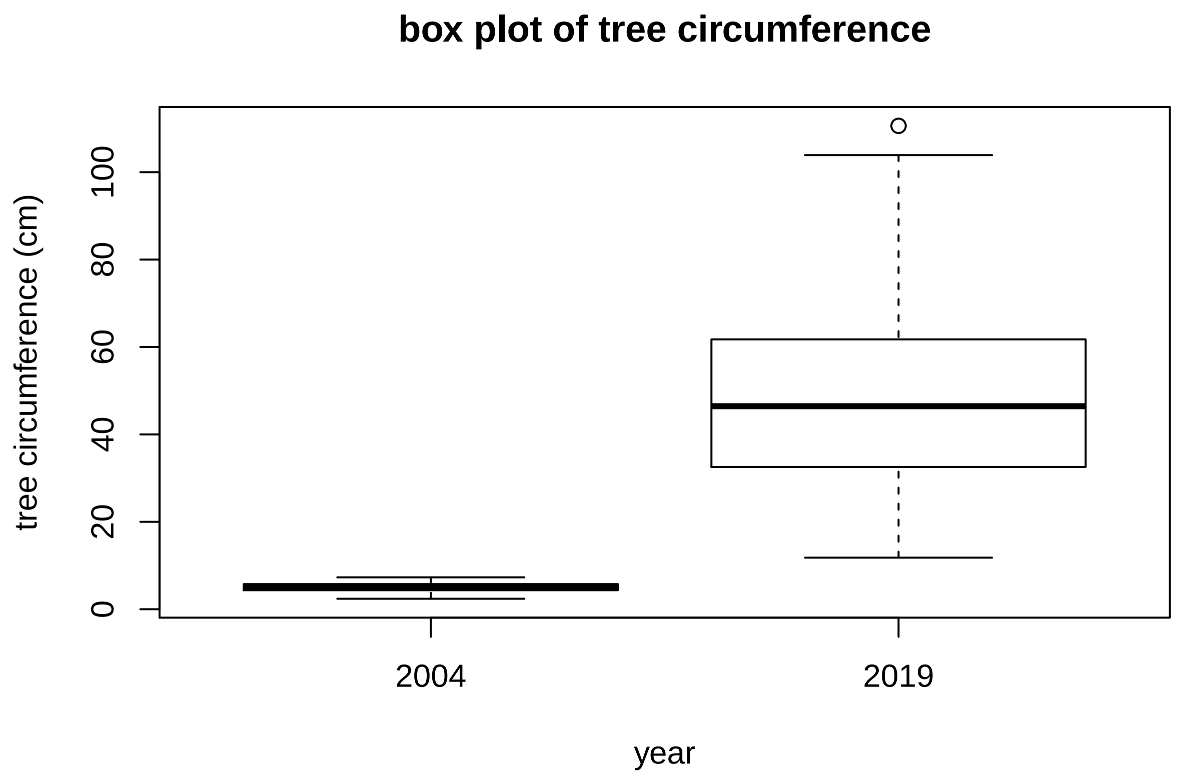
```
cat('SD at 2019: ', sd_2019)
```

```
## SD at 2019:  22.17979
```

Question 8

Using `boxplot` command to create a box plot of tree circumference at the start year(2004) and the end year (2019) at both sites.

```
boxplot(df$Circumf_2004_cm, df$Circumf_2019_cm,
    names=c("2004", "2019"), xlab = "year", ylab = "tree circumference (cm)", main = "box plot of tree
```



Question 9

Create a column growth (difference between 2009 and 2019) over the past 10 years for both site. Get growth values for each site and calculate mean. Mean growth over past 10 year for northeast and southwest are 30.076 cm and 48.354 cm respectively.

```
# calculate the difference of tree circumference between 2009 and 2019
df$growth <- df$Circumf_2019_cm - df$Circumf_2009_cm

# get growth values for each site
north_growth <- df[df$Site=="northeast", ]$growth
south_growth <- df[df$Site=="southwest", ]$growth

# calculate the mean for each site
mean_northeast <- mean(north_growth)
mean_southwest <- mean(south_growth)

cat("Mean growth of Northeast over the past  10 years: ", mean_northeast, '\n')
```

```
## Mean growth of Northeast over the past  10 years:  30.076
```

```
cat("Mean growth of Southwest over the past  10 years: ", mean_southwest)
```

```
## Mean growth of Southwest over the past  10 years:  48.354
```

Question 10:

Two functions `t.test` and `wilcox.test` are used to perform hypothesis tests (t test and wilcoxon test).p-value of t-test is 1.712524e-06 and p-value of wilcoxon test is 4.6264e-06.

```r
# run t-test
t_test_res <- t.test(north_growth, south_growth)
# take p-value
t_test_pvalue <- t_test_res$p.value

#run Wilcoxon test
wilcox_test_res <- wilcox.test(north_growth, south_growth)
# take the p-value
wilcox_pvalue <- wilcox_test_res$p.value

cat('p-value of t-test: ', t_test_pvalue, '\n')
```

```
## p-value of t-test:  1.712524e-06
```

```r
cat('p-value of wilcoxon test: ', wilcox_pvalue)
```

```
## p-value of wilcoxon test:  4.6264e-06
```

Part 2: Test the limits of BLAST

Thi Tran

5/31/2020

```r
# loading library
library(rBLAST)
```

```
## Loading required package: Biostrings

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit
```

```
library(Biostrings)
library(seqinr)
```

```
##
## Attaching package: 'seqinr'

## The following object is masked from 'package:Biostrings':
##
##     translate
```

```
source("https://raw.githubusercontent.com/markziemann/SLE712_files/master/bioinfo_asst3_part2_files/mut
```

Question 1:

The whole set of E.coli is downloaded by command `download.file`, uncompressed by `gunzip`, and creating a blast database by `makeblastdb()` function. There are 4,140 sequences present in the E.coli set.

```
# download the whole E.coli sequences
download.file("ftp://ftp.ensemblgenomes.org/pub/bacteria/release-42/fasta/bacteria_0_collection/escheri

# uncompress the file
R.utils::gunzip("Escherichia_coli_str_k_12_substr_mg1655.ASM584v2.cds.all.fa.gz", overwrite=TRUE)

# create the blast database
makeblastdb("Escherichia_coli_str_k_12_substr_mg1655.ASM584v2.cds.all.fa", dbtype="nucl", "-parse_seqid
```

Question 2:

The sample fasta sequences are downloaded by `download.file`, read in by `read.fasta`. My interest sequence is 13th sequence. `getLength()` and `GC()` function from `sequinr` library are used to get the length and the proportion of GC bases of my interest sequence. The length of this sequence is 273 bp and the GC proportion is 0.4908425.

```
# download sample fasta sequences
download.file("https://raw.githubusercontent.com/markziemann/SLE712_files/master/bioinfo_asst3_part2_fil

# read all sequences from fa file
all_sequences <-  seqinr::read.fasta("sample.fa")

# my id is 13, get sequences 13-th
my_seq <- all_sequences[[13]]

# get sequence length and calculate the GC content
length_seq <- seqinr::getLength(my_seq)
gc_proprotion <- seqinr::GC(my_seq)

#show the results
cat("Sequence length: ", length_seq, '\n')
```

```
## Sequence length:  273
```

```
cat("GC Proportion: ", gc_proprotion)
```

```
## GC Proportion:  0.4908425
```

Question 3

`myblastn_tab()` function is used to perform blast search on the whole E.coli sequence database. There are only two sequences that matches my sequence best. The first hit is my sequence itself and the second is a similar sequence.

```
# blast search for 13-th sequence
res <- myblastn_tab(myseq = my_seq, db = "Escherichia_coli_str_k_12_substr_mg1655.ASM584v2.cds.all.fa")

# top 3 hits including percent indentify, E-value and bit scores
head(res, 3)[,c("qseqid", "sseqid", "pident", "evalue","bitscore")]
```

```
##   qseqid   sseqid  pident    evalue bitscore
## 1     13 AAC73543 100.000 6.01e-150      525
## 2     13 AAC76974  77.966  1.15e-26      116
```

Question 4:

Using `mutator()` function to create mutated sequence with 20 point mutations and then compare with the original sequence by making a pairwise alignment by `pairwiseAlignment` from `Biostrings` library. The number of mismatch determined by `nmismatch()` function is 16.

```
# read my sequence under Biostring format
my_seq_new <- Biostrings::readDNAStringSet('sample.fa')[13]

# extract as a simple string
my_seq_str <- toString(my_seq_new)
```

```
# convert string to a vector of characters
my_seq_char <- seqinr::s2c(my_seq_str)

# create a mutated copy with 20 substitutions
my_seq_char_mut <- mutator(myseq=my_seq_char, 20)

## now create a pairwise alignment

# create DNAString format from mutation sequence
my_seq_mut <- DNAString(c2s(my_seq_char_mut))
aln <- Biostrings::pairwiseAlignment(my_seq_new, my_seq_mut)

# get the number of mismatch
num_mismatch <- nmismatch(aln)

# show the result
cat("Number of mismatches between the original and mutated sequence: ", num_mismatch)
```

```
## Number of mismatches between the original and mutated sequence:  12
```

Question 5:

Create a function `matching()` to make a n-point mutation sequence, run blast search and return result whether it identify matched sequences or not as a 0 or 1. Using `for` loop to run `matching()` for sequence having 1-273(the length of my interest sequence) point mutations, repeat 100 times for each sequence to get reliable results. Blast search reaches the limit when the number of sites need to be altered is 94, and the proportion of sites need to be altered is 0.3443223.

```
# create function for blast search of a mutated sequence
matching <- function(nmut) {
  db_path <- 'Escherichia_coli_str_k_12_substr_mg1655.ASM584v2.cds.all.fa'
  # create mutated sequence
  mut_seq_char <- mutator(my_seq_char, nmut)
  # convert to a vector of characters
  mut_seq <- seqinr::c2s(mut_seq_char)
  # create a table of blast search
  match_table <- (myblastn_tab(mut_seq, db_path))
  # check whether this table is null or not
  num_match <- dim(match_table)[1]
  if (is.null(num_match)) {
    return(0)
  }else{
    return(1)
  }
}
# calculate probability of finding a matched sequence for mutated sequences with the number of point mu

probs <- c()
for (nmut in 1:length_seq) {
  prob <- mean(replicate(100, matching(nmut)))
  probs[nmut] <- prob
}
```

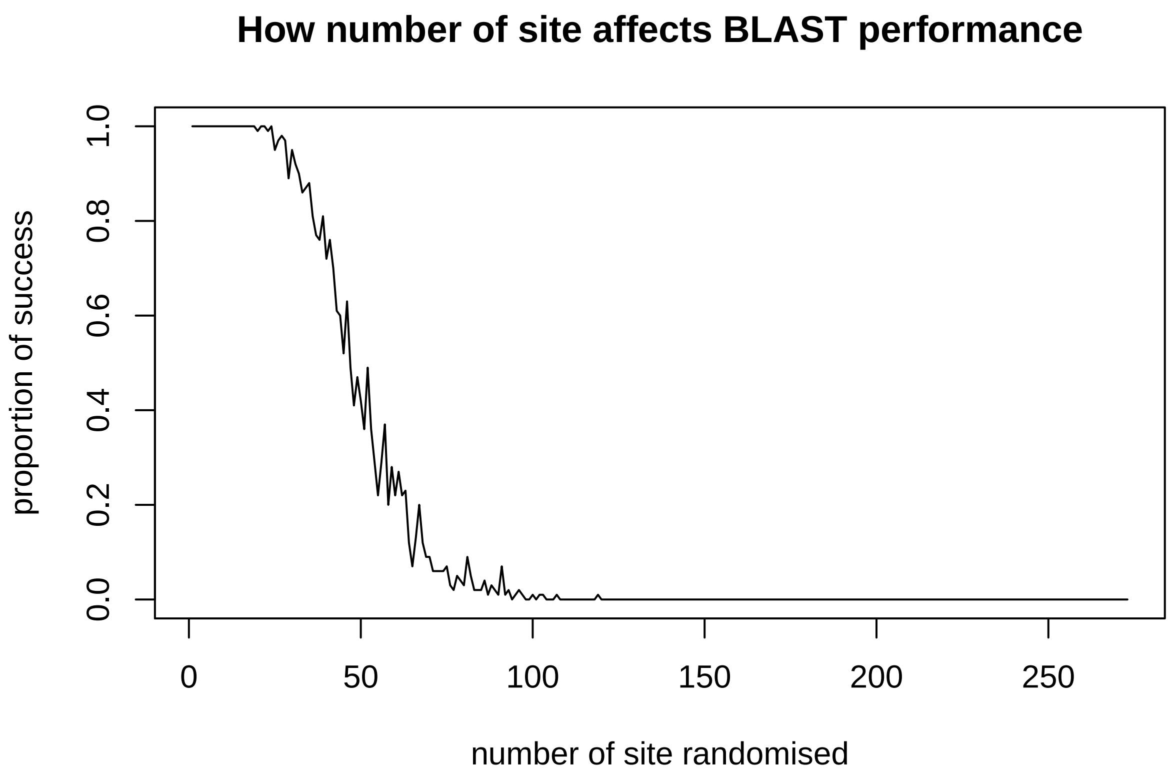
```
number_mutation <- seq(1:length_seq)

# get the first position that matching probability is zero
nmut_prevent_search <- which(probs == 0)[1]

# show the results
cat("the number of mutation that prevent BLAST search from matching: ", nmut_prevent_search)
```

```
## the number of mutation that prevent BLAST search from matching:  94
```

```
cat("the proprotion of mutation that prevent BLAST search from matching: ", nmut_prevent_search / lengt
```

```
## the proprotion of mutation that prevent BLAST search from matching:  0.3443223
```

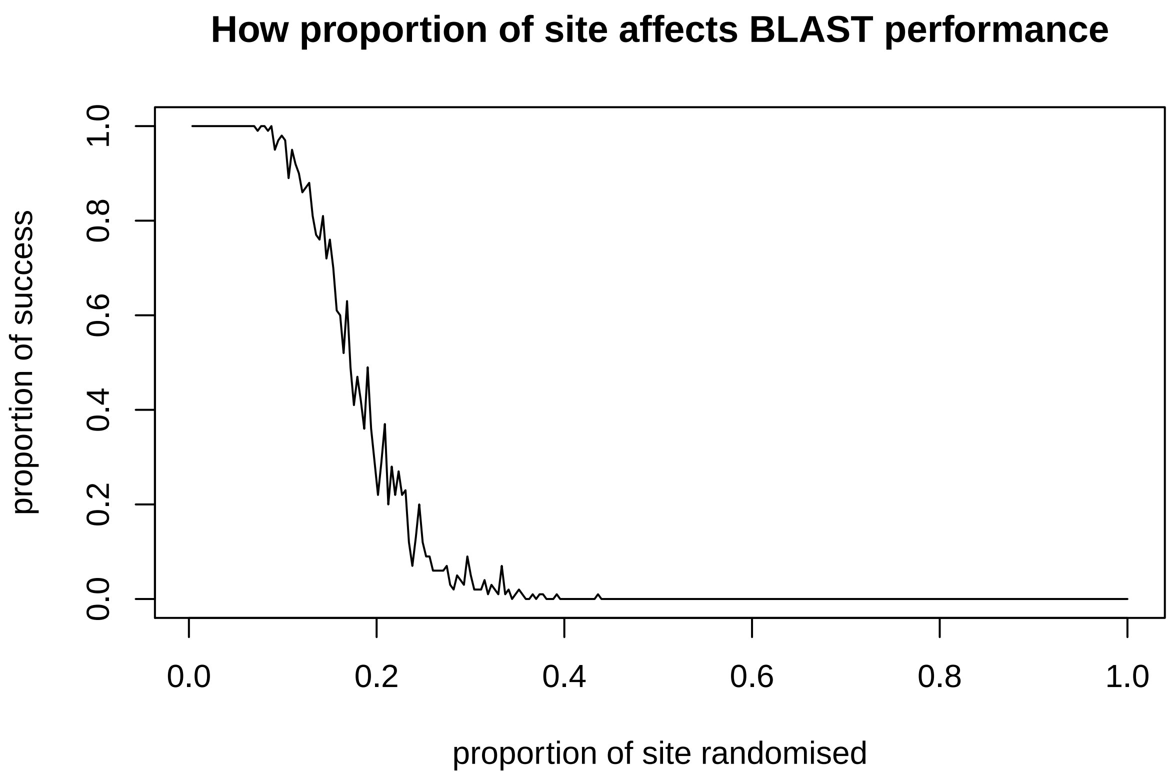
```
plot(number_mutation, probs, type='l', main = "How number of site affects BLAST performance", ylab = "p
```



Question 6:

Using `plot` to show the results. From the chart, the matching ability of blast decrease gradually from 100% to 0% with the proportion of sites from 10% to 40%. The ability for blast to match the gene of origin is 50% when the proportion of mutated sites at about 20%.

```
# calculaete matching probability following the proportion of altered sites
propotion_plot <- number_mutation / length_seq

# show the chart
plot(propotion_plot, probs, type='l', main = "How proportion of site affects BLAST performance", ylab =
```


How proportion of site affects BLAST performance
proportion of success
proportion of site randomised