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/mutb
```

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_seqids
```

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
}
```

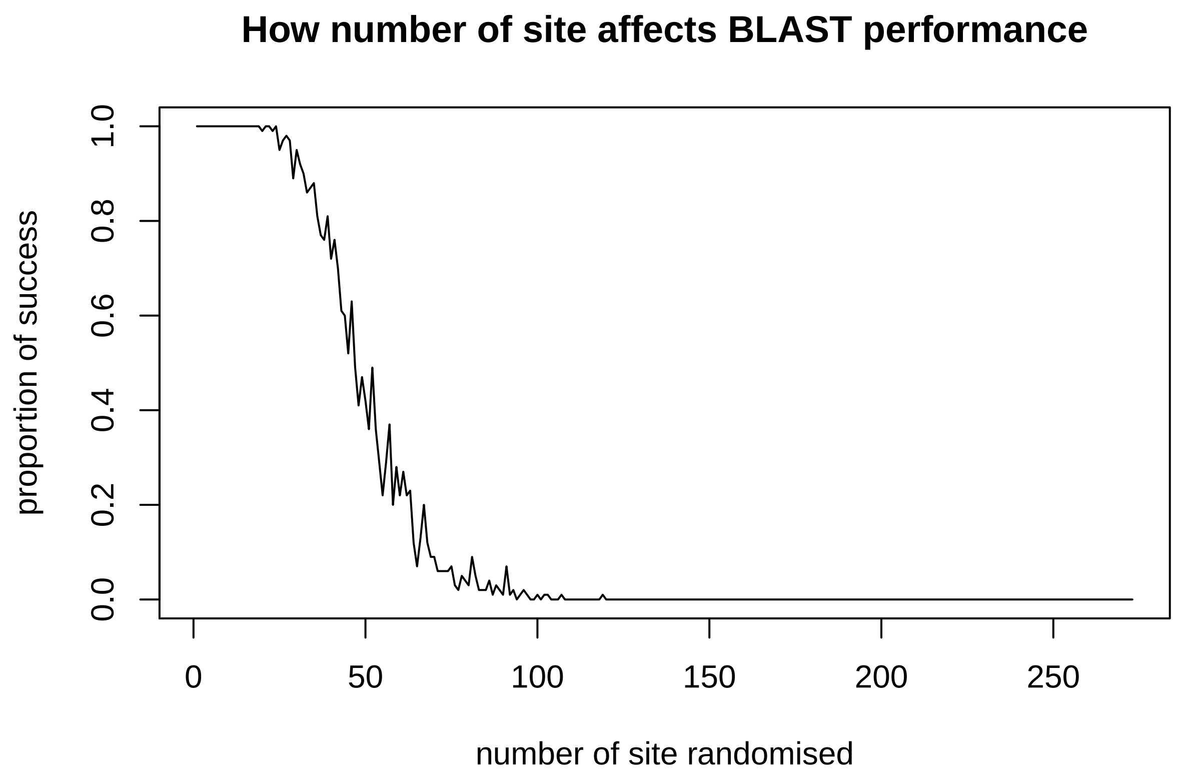
```r
number_mutation <- seq(1:length_seq)

# 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
```

```r
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
```

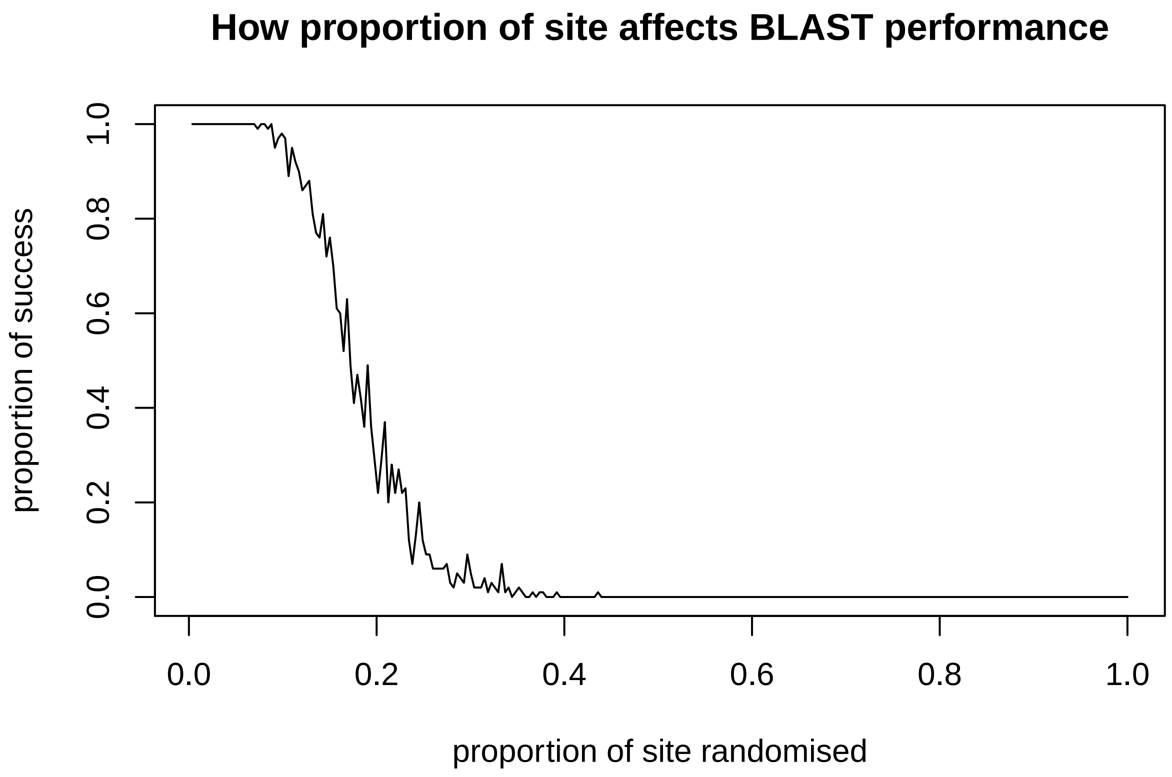
```r
plot(number_mutation, probs, type='l', main = "How number of site affects BLAST performance", ylab = "p
```



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%.

```r
# 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