HW8

Truc Huynh

4/11/2021

# Part A:

library(seqinr)  
  
# Source code for clean Alignment and print Multiple Alignment function  
source("cleanAlignment.R")  
source("printMultipleAlignment.R")

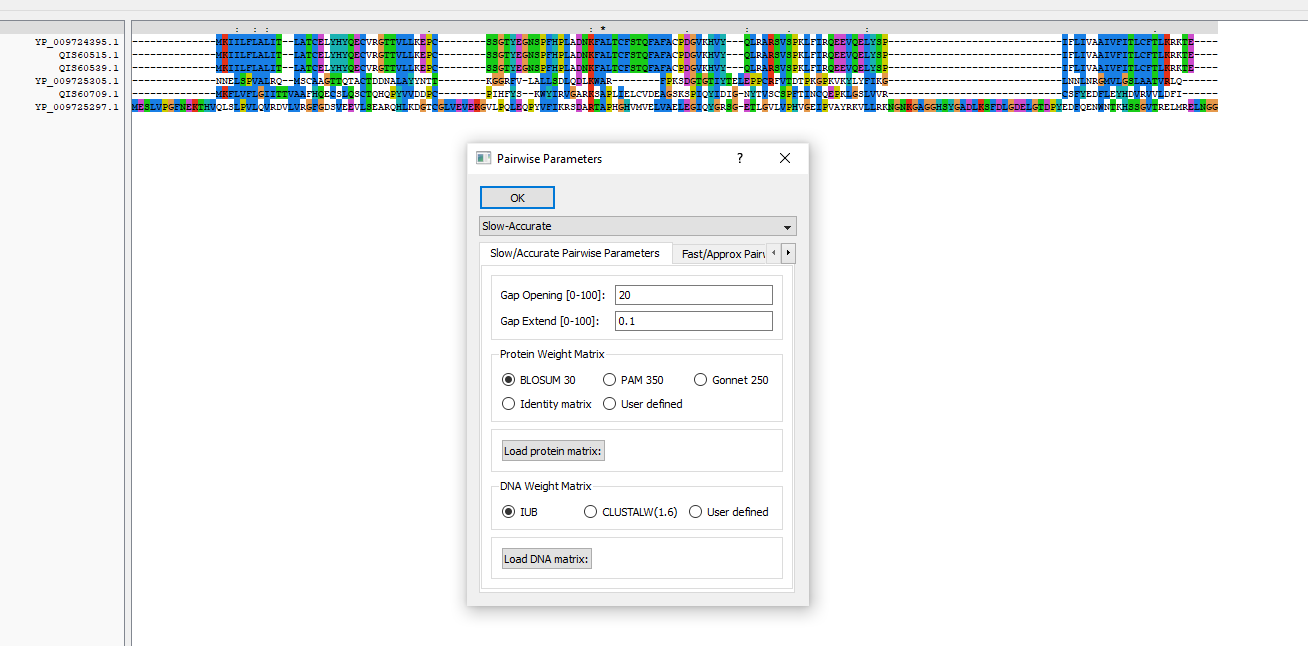
## Question 1

Retrieve the protein sequences with the following accession numbers: YP\_009725305, YP\_009724395, YP\_009725297, QIS60515, QIS60539, QIS60709

# Read the fasta file in to review it  
virus <- read.fasta("VIRUS.FASTA")

Instead of using swissprot, I will download all that protein sequences by using the NCBI website by the following steps: - Search for the protein sequences by accession number - Add each of them to the basket. - Download all when done in fasta file format - Open each file and put them together

## Question 2

Use CLUSTAL program to align the sequences. Adjust the pair-wise alignment parameters to use BLOSUM65 and a gap-opening penalty of 20. Provide a screen-shot of the result 

## Question 3

Read the alignment into R and compute the score of the alignment

# Read in Multiple Alignment Sequences  
obj<-read.alignment(file="VIRUS.phy", format = 'phylip')  
  
# Download all the column scores to file using ClustalX  
# Read the scores using read.table  
scores <- read.table(file="scores.qscores")  
  
# use the mean to calculate the average score  
mean(scores$V7)

## [1] 16.51934

## Question 4

Print the first 40 characters of the alignment.

# Print the first 40 characters of the alignmnet Object using Substring  
for (i in 1:6)  
{  
 print( substr(obj[["seq"]][[i]],1,40))  
}

## [1] "--------------mkiilflalit--latcelyhyqecv"  
## [1] "--------------mkiilflalit--latcelyhyqecv"  
## [1] "--------------mkiilflalit--latcelyhyqecv"  
## [1] "--------------nnelspvalrq--mscaagttqtact"  
## [1] "--------------mkflvflgiittvaafhqecslqsct"  
## [1] "meslvpgfnekthvqlslpvlqvrdvlvrgfgdsveevls"

cat("\n")

print("Using Multiple Alignment Printting with chunk size = 40")

## [1] "Using Multiple Alignment Printting with chunk size = 40"

cat("\n")

# Can also use the printMultipleAlignment  
# Use the print Multiple Alignment with chunksize = 40 to print the alignment to compare  
printMultipleAlignment(obj,40)

## 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.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

##   
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':  
##   
## windows

## Loading required package: XVector

##   
## Attaching package: 'Biostrings'

## The following object is masked from 'package:seqinr':  
##   
## translate

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

## [1] "--------------MKIILFLALIT--LATCELYHYQECV 24"  
## [1] "--------------MKIILFLALIT--LATCELYHYQECV 24"  
## [1] "--------------MKIILFLALIT--LATCELYHYQECV 24"  
## [1] "--------------NNELSPVALRQ--MSCAAGTTQTACT 24"  
## [1] "--------------MKFLVFLGIITTVAAFHQECSLQSCT 26"  
## [1] "MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLS 40"  
## [1] " "  
## [1] "RGTTVLLKEPC--------SSGTYEGNSPFHPLADNKFAL 56"  
## [1] "RGTTVLLKEPC--------SSGTYEGNSPFHPLADNKFAL 56"  
## [1] "RGTTVLLKEPC--------SSGTYEGNSPFHPLADNKFAL 56"  
## [1] "DDNALAYYNTT--------KGGRFV-LALLSDLQDLKWAR 55"  
## [1] "QHQPYVVDDPC--------PIHFYS--KWYIRVGARKSAP 56"  
## [1] "EARQHLKDGTCGLVEVEKGVLPQLEQPYVFIKRSDARTAP 80"  
## [1] " "  
## [1] "TCFSTQFAFACPDGVKHVY---QLRARSVSPKLFIRQEEV 93"  
## [1] "TCFSTQFAFACPDGVKHVY---QLRARSVSPKLFIRQEEV 93"  
## [1] "TCFSTQFAFACPDGVKHVY---QLRARSVSPKLFIRQEEV 93"  
## [1] "--------FPKSDGTGTIYTELEPPCRFVTDTPKGPKVKY 87"  
## [1] "LIELCVDEAGSKSPIQYIDIG-NYTVSCSPFTINCQEPKL 95"  
## [1] "HGHVMVELVAELEGIQYGRSG-ETLGVLVPHVGEIPVAYR 119"  
## [1] " "  
## [1] "QELYSP-----------------------------IFLIV 104"  
## [1] "QELYSP-----------------------------IFLIV 104"  
## [1] "QELYSP-----------------------------IFLIV 104"  
## [1] "LYFIKG-----------------------------LNNLN 98"  
## [1] "GSLVVR-----------------------------CSFYE 106"  
## [1] "KVLLRKNGNKGAGGHSYGADLKSFDLGDELGTDPYEDFQE 159"  
## [1] " "  
## [1] "AAIVFITLCFTLKRKTE---- 140"  
## [1] "AAIVFITLCFTLKRKTE---- 140"  
## [1] "AAIVFITLCFTLKRKTE---- 140"  
## [1] "RGMVLGSLAATVRLQ------ 132"  
## [1] "DFLEYHDVRVVLDFI------ 140"  
## [1] "NWNTKHSSGVTRELMRELNGG 199"  
## [1] " "

## Question 5

Write a function to find the longest stretch of the complete conserved positions in the alignment. Test your function on the alignment of question A-3. Hint: “complete” means 100% identical letters.

LongestStretch <- function(obj){  
 cleanAlignment(obj,100,100)  
}

# Call Clean Alignment and set the minimum identical letter to 100  
LongestStretchAln<- LongestStretch(obj)   
  
printMultipleAlignment(LongestStretchAln)

## [1] "A 60"  
## [1] "A 60"  
## [1] "A 60"  
## [1] "A 60"  
## [1] "A 60"  
## [1] "A 60"  
## [1] " "

# Part B:

## Question 1

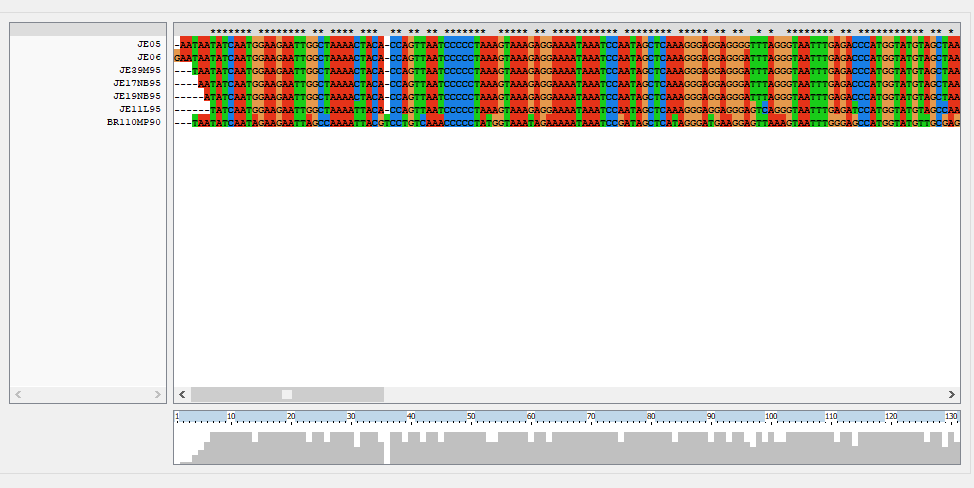
Read the sequence data and store them in a FASTA file

There is 2 way to do this: ### Using Clusatx: - Read the .txt file into clustalX. - Save the file as a .FASTA file (export) ### Using R

data <- read.table("~/source/R\_for\_Bioinformatic/R\_Bioinformatic\_Basic/L10/data.txt", quote="\"", comment.char="")  
  
# Check the dimension and overview of the data  
View(data)  
  
# Write the file to the fasta file  
write.fasta(list(data[2,],data[4,],data[6,],data[8,],data[10,],data[12,],data[14,]),list(data[1,],data[3,],data[5,],data[7,],data[9,],data[11,],data[13,]),file.out = "dataR.fasta")

Both methods return the same result

## Question 2

Use CLUSTAL program to align the sequences. Use a penalty for a gap opening equals 15 and a gap extension to be 3 

## Question 3

Read the alignment in R and print the first 20 positions of the alignment for only the first 3 sequences

# Read in Multiple Alignment Sequences  
objAlignment<-read.alignment(file="data.phy", format = 'phylip')  
  
# Print the first 20 characters of the alignment Object (first 3 sequences) using Substring  
for (i in 1:3)  
{  
 print( substr(objAlignment[["seq"]][[i]],1,20))  
}

## [1] "-aataatatcaatggaagaa"  
## [1] "gaataatatcaatggaagaa"  
## [1] "---taatatcaatggaagaa"

## Question 4

Write a function to filter the alignment and keep only poorly conserved regions. Your function should accept two parameters: the alignment object, and the minimum percent of letters in an alignment column that must be gap characters for the column to be kept.

# the poorlyConservedVirusaln loop through each rows and compare the'-' value in the column.  
# Find the gap percentage in each column an compare it to the minimum gap percentage.  
# Store the column that have its percentage to the newAlignment Object seq and return it.  
  
poorlyConservedVirusaln<-function(alignment,minpcgap) {  
   
 # make a copy of the alignment to store the new alignment in:  
 newalignment <- alignment  
   
 # find the number of sequences in the alignment  
 numseqs <- alignment$nb  
   
 # empty the alignment in "newalignment")  
 for (j in 1:numseqs) { newalignment$seq[[j]] <- "" }  
   
 # find the length of the alignment  
 alignmentlen <- nchar(alignment$seq[[1]])  
   
 # look at each column of the alignment in turn:  
 for (i in 1:alignmentlen)  
 # Travel through the length of the alignment  
 {  
 # see what percent of the letters in this column are non-gaps:  
 gap <- 0  
 for (j in 1:numseqs)  
 {  
 seqj <- alignment$seq[[j]]  
 letterij <- substr(seqj,i,i)  
 if (letterij == "-") { gap <- gap + 1}  
 }  
 pcgap <- (gap\*100)/numseqs  
 # Only consider this column if at least minpcnongap % of the letters are not gaps:  
 if (pcgap >= minpcgap)  
 {  
 # see what percent of the pairs of letters in this column are identical  
 for (j in 1:numseqs)  
 {  
 seqj <- alignment$seq[[j]]  
 letterij <- substr(seqj,i,i)  
 newalignmentj <- newalignment$seq[[j]]  
 newalignmentj <- paste(newalignmentj,letterij,sep="")  
 newalignment$seq[[j]] <- newalignmentj  
 }  
 }  
 }  
 return(newalignment)  
}

## Question 5

Test the function in part (B-4) using the alignment in part (B-2) considering at least 75% of gap in a position. Comment on the output

notcleanedvirusaln <- poorlyConservedVirusaln(objAlignment, 75)  
printMultipleAlignment(notcleanedvirusaln)

## [1] "-----T-- 53"  
## [1] "G-----A- 54"  
## [1] "-------- 52"  
## [1] "-------- 52"  
## [1] "---AA--- 54"  
## [1] "--G----G 54"  
## [1] "-T------ 53"  
## [1] " "

## Question 6

Find the genetic distance of the alignment

# The generic distance of the alignment can be calculate using   
dist.alignment(objAlignment)

## JE05 JE06 JE39M95 JE17NB95 JE19NB95 JE11L95   
## JE06 0.11009638   
## JE39M95 0.09016696 0.11043153   
## JE17NB95 0.07808688 0.10091234 0.04517540   
## JE19NB95 0.12751534 0.12764513 0.12777531 0.12725695   
## JE11L95 0.20680875 0.21189139 0.19185884 0.20617983 0.17996851   
## BR110MP90 0.38441290 0.38480258 0.37719547 0.38177086 0.38703691 0.38215785