SARS-CoV-2 Genomic Analysis

Final Project – Spring 2020

BI-GY 7663 – Professor Mgavi Brathwaite
By: Lindsay Reisman

Introduction

2019-nCoV has a striking similarity to SARS-CoV but what makes it stand out is how easily it is transmitted between humans. It creates a dense glycosylated spike (S) protein which allows the virus to invade the cell. Spike proteins tend to form trimers around the viral surface, creating a crown like appearance which is where the Coronavirus name was derived. The spike protein binds to the receptor, helping facilitate the protein to undergo a structural conformational change by fusing the viral and host membranes. ACE2 is the primary receptor for 2019-nCoV, the spike protein has a higher affinity for the virus to bind to the enzyme. This enzyme is crucial in regulatory functions found in various cell types in all major organs such as the heart and lungs. For this project we are focusing on ORF1ab polyproteins and their role in 2019-nCoV. These polyproteins are translated from ORF1a and ORF1b ribosomal frameshift. ORF1ab is the driving force for Covid-19 by viral replication.

Preprocessing

The data must be preprocessed before we aligned the sequences with Clustal Omega. For preprocessing pipeline, a lot of skills learned in the beginning of the semester such as functions, parsing files, dictionaries and regular expressions were used to process the data to use as an input file for the Multiple Sequence Alignment (MSA). Other tools were imported such as import re for regular expressions, from bio import Entrez for querying locus IDs, and import os to allow interactions with the operating system (os). The SNP mutations were mapped to the gene strains using a function to iterate over all the gene strains created from a previous assignment. Once the mutations were mapped, using Entrez from Bio Python, the locus id's were queried to grab the translated coding sequences for the input file.

Please see the appendix for commented code.

Summary of MSA analysis

After the necessary commands of ./configure, make, sudo make install and make clean to download the package for Clustal Omega and argtable to my local machine. I was able to run the MSA, using command line (as seen below):

Clustalo -i /directory path to file/ - o /directory for outfile -outfmt=clu -v to run

```
(base) Lindsays-MacBook-Pro:clustal-omega-1.2.4 lindsayreisman$ clustalo -i /Users/lindsayreisman/Documents/Final\ Project/Translated\ Data/ORF1AB/ORF1AB_mutatedPP.fasta -o /User
s/lindsayreisman/Documents/Final\ Project/aligned.msa -outfmt=clu -v
WARNING: Unsupported old command line option '-outfmt=clu' will be ignored (may change default output stream and format)
WARNING: Your old-style command-line options were converted to:  clustalo -i /Users/lindsayreisman/Documents/Final Project/Translated Data/ORF1AB/ORF1AB_mutatedPP.fasta -o /Users
/lindsayreisman/Documents/Final Project/aligned.msa -v --outfmt=clustal -v --force
Using 1 threads
seq-type = UNKNOWN
seq-in-fmt = unknown
option: seq-in = /Users/lindsayreisman/Documents/Final Project/Translated Data/ORF1AB/ORF1AB_mutatedPP.fasta
option: dealign = 0
option: profile1 = (null)
option: profile2 = (null)
option: is-profile = 0
option: max-num-seq = 2147483647
option: max-seq-len = 2147483647
option: aln-out-file = /Users/lindsayreisman/Documents/Final Project/aligned.msa
option: aln-out-format = Clustal
option: force-file-overwrite = 1
option: line wrap = 60
option: print residue numbers = 0
option: order alignment like input/tree = 0
option: threads = 1
option: PseudoFile = (null)
option: logFile = (null)
option: auto-options = 0
option: distmat-infile = (null)
option: distmat-outfile = (null)
option: clustering-type = 1
option: pair-dist-type = 1
option: use-mbed = 1
option: use-mbed-for-iteration = 1
option: pile-up = 0
option: guidetree-outfile = (null)
option: guidetree-infile = (null)
option: hmm-input-files = 0
option: num-iterations = 0
option: iterations-auto = 0
option: max-hmm-iterations = 2147483647
option: max-guidetree-iterations = 2147483647
option: iMacRamMB = 8000
option: percent-id = 0
option: use-kimura = 0
option: clustering-out = (null)
option: posterior-out = (null)
```

From the differences between the sequences we can analyze the evolution of SARS/CoV-2. We can compare similar sequences with large gaps and explore the differences from the reference. The gaps relate to the protein structure and the presence of the spike protein and how ultimately the virus affected the cells which is why the virus is so variable from host to host.

Appendix

```
import re
import os

Step 1: Map SNPs
 #open file and readlines #to turn the string into list
file1 = open("/Users/lindsayreisman/Documents/NC_045512.fasta", 'r')
r = file1.readline() #skip first line
```

```
NC_045512_list = [b for line in file1.read().split('\n') for b in line]
file1.close()

#open the count file for SNP coordinates
file2 = open("/Users/lindsayreisman/Documents/countResult.txt", "r")
file2.readline() #skip first line
count_dict = {} #create a dictionary for later to map SNP's to all the strains

for line in file2.readlines():
    coordinate, change = line.split(":") #split the SNP coordinates by ':' separating the coordinate
#from the change
    count_dict[int(coordinate)] = [change[1],change[4]] #[num1,num2] #creating keys for dict
    NC_045512_list[int(coordinate) - 1] = change[4]   # write out the first account for the -1
counting, the change needed is the 4th index
#print(count_dict)
file2.close()

NC_045512_mutated = "".join(NC_045512_list) #using the join method creates a string of the
mutated fasta.

with open("/Users/lindsayreisman/Documents/Final Project/mutated_NC_045512.fasta.txt", "w")
as f:
    f.write(r) #header
    f.write(NC_045512_mutated) #write the fasta sequence to the file created


# Next map the rest of the strains to the SNP changes in the count result file using function
#iterate through each strain output file (from previous assignment and map to #mutated sequence
def map_strain():
    g = ['E','N','orf1ab','S','M','ORF10','ORF3a','ORF8'] #define variable
    directory = '/Users/lindsayreisman/Documents/Final Project/'
    for gene in g:
        # make directory for each gene
        os.mkdir(directory + '/Mutated Gene Strains/' + gene)
        for i in os.listdir(directory+gene): #using the files from previous assignment, list all the
##strains within the directory for reading
            #print(i)
            files = open(directory+gene+'/'+i) #open the files
            r = files.read() #read the files

            #split the string on the newline to separate the seq from the definition line
            defline = r.split('\n')[0]

            # define the seq by joining r and capture everything in the list after the second index (1)
            seq = ''.join(r.split('\n')[1:])
```

```
            range1= int(r.split('-')[0].split(':')[2]) # define range to map the SNP coordinates to
            range2= int(r.split('-')[1].split('|')[0])
            #print(seq)
            #print(range1,range2)
            # print(seq[range1-range1])

            #create file to write out to
            out = open(directory + '/Mutated Gene Strains/' + g + '/mutated_'+i, "w")
            for x in range(range1, range2): # loop through to find the range
               # print(seq[x-range1])

               if x in count_dict:# now loop through to the dictionary to map to SNP coordinates
                  if seq[x-range1-1] == count_dict[x][0]: # subtract int from range so index 2500
#would be read as zero and then -1 to account for the zero indexing
                     seq = seq[:x-range1-1] + count_dict[x][1] + seq[x-range1:]
                     #print(x-range1-1,count_dict[x], seq[x-range1-1:])
                  #print(count_dict[x])

            out.write('>'+defline+'\n'+seq) #write to file definition line and sequence
            #print(seq)

      map_strain() #call the function
```

Step 2: Parse the coding sequences for all ORF1ab genome strains

```
content = ' '
directory = '/Users/lindsayreisman/Documents/Final Project/Mutated Gene Strains/orf1ab/'
#create directory
for i in os.listdir(directory): #list directory used to iterate over all the files within directory
   files = open(directory+i) #open the files within directory
   #print(i)
   r = files.read() #read the files
   content += r + '\n'+'\n'

out = open("/Users/lindsayreisman/Documents/Final Project/ORF1ab_CDS.fasta",'w')
out.write(content)
```

Step 3: Translate the sequences

```
from Bio import Entrez
import re
import os
Entrez.email = "ljr390@nyu.edu"
content = ' '
with open("/Users/lindsayreisman/Documents/Final Project/ORF1ab_CDS.fasta", "r") as handle:
   genes = handle.read() #read handle
   all _genes = genes.split('\n\n')
   that_gene in all _genes[:-1]:
```

```
    n_split = that_gene.split(':') #split based on colon
    #print(n_split[:10])
    locus = n_split[1] #capture the second index to query in Entrez – the locus locator
    #bp_len = that_gene.split('-')[0]
    #print(bp_len)
    #use efetch to grab the PP translation from the genbank
    handle = Entrez.efetch(db ="nucleotide", id= locus  ,rettype="gb", retmode="xml")
    record = Entrez.read(handle)
    #print(gene,seq,locus)
    #for r in record[0]:
    #    print(r)

    #dicitonaries within dictionaries – looped through to gather the information needed.
    feature_table = record[0]['GBSeq_feature-table'] # find the data to use in the in the directory
    for f in feature_table:
        if f['GBFeature_key'] == 'CDS':
            #print(f)

            for G in f['GBFeature_quals']:
                if G['GBQualifier_name']== 'gene':
                    gene_v1 = G['GBQualifier_value'].upper()
                if G['GBQualifier_name']== 'translation':
                    gene_v2 = G['GBQualifier_value']
                if G['GBQualifier_name']== 'protein_id':
                    gene_v3 = G['GBQualifier_value']
                if G['GBQualifier_name']== 'product':
                    gene_v4 = G['GBQualifier_value']

            content +='>' + gene_v3+ '\n' + gene_v2 + '\n' + '\n'


gene_file = "/Users/lindsayreisman/Documents/Final Project/ORF1ab_PP.fasta" #create an
output to write to

out = open(gene_file, 'w')
out.write(content)
```

Reference

1. Sandrine Belouzard, *et al.* (2012) Mechanisms of Coronavirus Cell Entry Mediated by the Viral Spike Protein. Viruses. 4(6): 1011–1033.
   Taylor Heald-Sargent, *et al.* (2012) Ready, Set, Fuse! The Coronavirus Spike Protein and Acquisition of Fusion Competence. Viruses. 4(4): 557–580.
   Lanying Du, *et al.* (2009) The spike protein of SARS-CoV — a target for vaccine and therapeutic development. Nat Rev Microbiol. 7(3): 226–236.
   Xu, Y., *et al.* (2004) Crystal structure of severe acute respiratory syndrome coronavirus spike protein fusion core. J.Biol.Chem. 279: 49414-49419.

2. Wrapp D, Wang N, Corbett KS, et al. Cryo-EM structure of the 2019-nCoV spike in the prefusion conformation. *Science*. 2020;367(6483):1260-1263. doi:10.1126/science.abb2507
3. Graham, Rachel L et al. “SARS coronavirus replicase proteins in pathogenesis.” *Virus research* vol. 133,1 (2008): 88-100. doi:10.1016/j.virusres.2007.02.017
4. Woo PC, Huang Y, Lau SK, Tsoi HW, Yuen KY. In silico analysis of ORF1ab in coronavirus HKU1 genome reveals a unique putative cleavage site of coronavirus HKU1 3C-like protease. *Microbiol Immunol*. 2005;49(10):899-908. doi:10.1111/j.1348-0421.2005.tb03681.x