ace2-dbsnp-results

August 11, 2020

1 SARS-2 and Human ACE2 Variant Analysis

1.1 Purpose

The intent of this pipeline is to identify SNVs in publically available, GISAID submitted SARS-2 genomes that differ from the original Wuhan strain. Identified amino acid variants in the S protein were then modeled for their 3D structure and interactions were predicted agains the Human ACE2 receptor and its reported variants.

1.1.1 1. What are the mutations within the SARS-2 genome and how prevalent are they?

In order to identify the mutations that are in circulation we can use the SARS-2 genomes from GISAID and compare them to the established Wuhan strain (the only RefSeq in NCBI).

A caveat: I do not know how these genomes are generated but they are presumably MinION data so performing traidional QC is not possible.

From there I can use the MUMMER4 package to identify mutations and their subsequent effect on translation.

1.1.2 2. Do any of these mutations have an effect on the binding affinity to ACE2?

Hin Hark is using these mutations in his modeling. I will ask him for a summary of his methods.

1.1.3 3. What are the mutations within the ACE2 receptor and do they have any effect on the binding affinity to the SARS-2 spike (s) protein?

Data for the mutations were obtained from dbSNP and is using the data from the sources mentioned below.

1.2 SNP Analysis

To start we are looking only at the spike protein. Using the SARS-2 reference sequence from NCBI I aligned all of the GISAID CDS in protein-space to the reference sequence. I then filtered the alignments to only the first reading frame and removed the ambiguous bases in the resulting SNPs.

```
[ ]: %%bash
# Format sequence identifier to play nice with Promer
perl -pe 's/^.+EPI_ISL_(\d+).+/>$1/g'  gisaid_hcov-19_2020_05_04_17.fasta |␣
 ↪perl -pe 's/-/n/g' > gisaid_hcov-19_2020_05_04_17.refactored.fasta

promer -p SARS-2.refseq.cds.s_prot.promer SARS-2.refseq.cds.s_prot.fasta␣
 ↪gisaid_cov2020_sequences.refactored.fasta
show-coords -clT SARS-2.refseq.cds.s_prot.promer.delta > SARS-2.refseq.cds.
 ↪s_prot.promer.coords
show-snps -STH gisaid_hcov-19_2020_05_04_17.s_prot.promer.delta < <(awk '$14 ==␣
 ↪"1" && $15 == "1" { print $0 }' gisaid_hcov-19_2020_05_04_17.s_prot.promer.
 ↪coords) | cut -f1-3 | sort | uniq -c | sort -nrb | perl -pe 's/ +/\t/g' |␣
 ↪perl -pe 's/^\s+//g' > gisaid_hcov-19_2020_05_04_17.s_prot.promer.
 ↪snps-with-nonsense-mutations.counts.tsv
# Removes nonsense mutations
# show-snps -ST SARS-2.refseq.cds.s_prot.promer.delta < <(awk '$14 == "1" &&␣
 ↪$15 == "1" { print $0 }' SARS-2.refseq.cds.s_prot.promer.coords) | grep -v␣
 ↪'X' > SARS-2.refseq.cds.s_prot.promer.snps
```

Parsed the dbSNP into a TSV file with the following command:

```
[ ]: # Added this into the show-snps one liner above
# perl -pe 's/\n$/\t/g' snp_result.txt| perl -pe 's/--\t/\n/g' | perl -pe 's/
 ↪\d+. rs/\nrs/g' | perl -pe 's/\r//g' > snp_result.tsv
```

From there I formatted the data into an extended TSV by adding the frequencies of the mutation per data source and extracted the other relevant info (discarding much of it data provided by dbSNP that I deeded irrelevent at this stage in the analysis).

```
[54]: snps = []

import re

def check_snp_src(l, src):
    index = [i for i, s in enumerate(l) if src in s if i is not ""]
    if index:
        m = re.search(r'0\.\d+',l[index[0]])
#           print(l[index[0]])
#           print (m.group(0))
#           print(m)
        if m:
            return m.group(0)
        else:
            return 'NA'
    else:
        return 'NA'
```

```
with open('snp_result.tsv', 'r') as f:
    for line in f.readlines():
        if line == '\n':
            continue
        snp = {
                'id':'',
                'snv':'',
                'position':'',
                '1000Genomes':'',
                'TWINSUK':'',
                'GnomAD':'',
                'ALSPAC':'',
                'TOPMED':''
              }
        l = line.split('\t')
        snp['id'] = l[0]
        snp['snv'] = l[1]
        snp['position'] = l[2]
        snp['1000Genomes'] = check_snp_src(l,'1000Genomes')
        snp['TWINSUK'] = check_snp_src(l,'TWINSUK')
        snp['GnomAD'] = check_snp_src(l,'GnomAD')
        snp['ALSPAC'] = check_snp_src(l,'ALSPAC')
        snp['TOPMED'] = check_snp_src(l,'TOPMED')

#         print(snp)

        snps.append(snp)

# with open('snp_result.cleaned.tsv', 'wb', )
df = pandas.DataFrame(snps)
df.to_csv('snp_result.cleaned.tsv', sep='\t', )

# print (snps)

#         for i in l:
#             if i == l[0]:
#                 snp['id'] = l[0]
#                 continue
#             elif i == l[1]:
#                 snp['snv'] = l[1]
#                 continue
#             elif i == l[2]:
#                 snp['position'] = l[2]
#                 continue
#             elif
```

[ ]:
```

```