Lab 7: Phylogenetics and COVID-19

Skills: Multiple sequence alignment, Phylogenetics, Installing tools, Finding data

This lab is optional, but you can earn extra credit (up to 2% of the final grade) by completing it. Note, there is one report notebook and the readme to fill out.

Intro

COVID-19 is caused by a novel type of coronavirus, a class of viruses that can cause disease ranging from the common cold to more severe conditions like MERS or SARS. In this lab, we'll analyze the genome multiple strains of the novel coronavirus (SARS-CoV-2), and use comparative genomics techniques to explore how the virus relates to other types of coronaviruses.

Our primary goal will be to reconstruct the phylogenetics analysis from A Novel Coronavirus from Patients with Pneumonia in China, 2019, one of the earliest reports of the full SARS-CoV-2 genome. Specifically, we'll be producing something like their Figure 4b:

This tree shows the relationship between multiple SARS-CoV-2 genomes and other coronavirus strains, including SARS, MERS, and coronavirus strains isolated from bats.

Unfortunately, they don't give us a whole lot of methods info to go on. Here is a snippet from the relevant methods part: "Multiple-sequence alignment of the 2019-nCoV and reference sequences was performed with the use of Muscle. Phylogenetic analysis of the complete genomes was performed with RAxML (13) with 1000 bootstrap replicates and a general time-reversible model used as the nucleotide substitution model." Not a whole lot of details, but with enough detective work we'll still be able to construct a similar tree!

Overview

Our overall goal will be to compare genomes of different virus species. Whereas in previous labs, we usually started from raw reads (in fastq format), for this lab, we'll work with existing assemblies for the viral genomes of interest and skip doing the assembly ourselves. If you choose to do the extra credit part, you'll try your hand at starting from the raw reads and using your own assembly in the phylogenetics analysis.

In this lab, we'll go through:

1. Obtaining genome assemblies from NCBI and raw reads from SRA.
2. Performing multiple sequence alignment.
3. Building a phylogenetic tree to explore the evolutionary relationship between virus strains.
4. Visualizing phylogenetic trees.

Summary of tools covered

In this lab we'll be using or referring to the following tools:

- mafft: for performing multiple sequence alignment
- RaxML: for building phylogenetics trees

1. Downloading the data

Our first step will be to download the assembled genomes for the viruses we'd like to compare. To help you get started, we've provided (based on manually copying from the figure above...) the NCBI accession numbers for the genomes to compare:

```
~/public/lab7/lab7_accessions.txt
```

This is just a text file, with one accession per line, where each accession is unique to a virus strain. These accessions, and brief descriptions, are also listed here: https://docs.google.com/spreadsheets/d/1p1JpKKj1lUmGqrq2fdnX-FHvv7wt-jIdyY4qrRXyT9I/edit?usp=sharing

To see info about a certain accession, you can go to NCBI using a link like: https://www.ncbi.nlm.nih.gov/nuccore/AY508724.1

You'll see for instance that this genome is from "SARS coronavirus NS-1, complete genome" (the original SARS from the early 2000s). You can also scroll through the entire genome sequence since it's so short!

To see another accession, replace the last value after the "/" in the URL with the genome you're interested in.

Our goal in this first section is to download the genomes for each of these accessions into one big fasta file, which we will need to input into the tools we use below. So you'd like to create a fasta file that looks something like this:

```
>AY508724.1 SARS coronavirus NS-1, complete genome
TACCCAGGAAAAGCCAACCAACCTCGATCTCTTGTAGATCTGTTCTCTAAACGAACTTTAAAATCTGTGT...
>AY485277.1 SARS coronavirus Sino1-11, complete genome
ATATTAGGTTTTTACCTACCCAGGAAAAGCCAACCAACCTCGATCTCTTGTAGATCTGTTCTCTAAACG...
```

Put your fasta file in `~/lab7/lab7_virus_genomes.fa`.

Some things you might find helpful:

- URLs of the form: https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&id=KC545386.1&rettype=fasta&retmode=text (similar for other accessions) will point you to the genome sequence for each accession (in fasta format)
- You can use the commands `wget` or `curl` to directly download files from URLs.
- Note: in some cases the downloaded sequences have extra empty lines at the end. You should remove empty lines in your final fasta file, either using UNIX or python (or any other method) before moving on.

A good way to go about this would be to write a for loop or similar script that would loop through accessions, construct a URL to the fasta file for each, then use `wget` to download. Our solution is able to do all of this in just one line of UNIX commands. Although your solution doesn't have to.

2. Performing multiple sequence alignment

Now that we've collected all the genome sequences we need in a single fasta file, we'll want to compare them to each other. A typical and critical step in order to do so is to create a "multiple sequence alignment" between them so we can compare nucleotides at specific bases across strains.

The NEJM paper used a tool called MUSCLE for this. We'll deviate from their methods and use an alternative tool called mafft, since we found that it runs quite a bit faster. `mafft` (and `MUSCLE`) take as input a fasta file with multiple genomes, and outputs a new fasta-like file showing the alignment between those genomes.

2.1 Install mafft

You'll first need to install `mafft`. Since we don't have "root" permissions, you'll want to follow their instructions here: https://mafft.cbrc.jp/alignment/software/installation_without_root.html to install to your own home directory.

Some notes that may be helpful:

- If you change the top line of the Makefile to `PREFIX=$(HOME)/local`, it will install to your home directory
- This will install `mafft` to `$HOME/local/bin`. You can run on the command line with the full path: `$HOME/local/bin/mafft`.
- Alternatively, you can add this directory to your `$PATH`, which is where UNIX searches for tools. For instance if you do

```
export PATH=$PATH:$HOME/local/bin
```

You should then be able to just type `mafft` at the command line like you would for other tools. Before moving on make sure when you type `mafft` or `$HOME/local/bin/mafft` in the command line the mafft instructions come up.

Note, if you close and reopen your terminal, you'll have to type this line again to reset the $PATH.

2.2 Run mafft

Once mafft is installed, you'll need to run it! If you type the `mafft` command, it will walk you through inputting the fasta file and ask a name for our output file. Run `mafft` and save the output to a file `lab6_virus_genomes.aln`. Look at the output file. It should have many gap "-" characters in addition to nucleotides in it. This so that all the comparable nucleotides are lined up in the final multiple sequence alignment.

3. Building a tree

Now, we're ready to build our tree. We'll use a tool called RaxML (the same one used in the NEJM paper). You can find the RaxML manual here: https://github.com/stamatak/standard-RAxML/blob/master/manual/NewManual.pdf. If the github page doesn't render the pdf properly in your web browser, you can download the pdf and view it locally. There is a lot of stuff in this manual, so please do not try to read the whole thing. We'll give some guidance in how to use this tool to build our tree.

3.1 Install RaxML

You'll first have to install RaxML. We downloaded the `.tar.gz` file for v8.2.12 from Github: https://github.com/stamatak/standard-RAxML/tags and followed the instructions on their Github home page (https://github.com/stamatak/standard-RAxML) for compiling the version "SSE3.PTHREADS". After it compiles, it should create a binary file `raxmlHPC-PTHREADS-SSE3`. It is convenient to add to a location on your $PATH, e.g.:

```
cp raxmlHPC-PTHREADS-SSE3 ~/local/bin/raxml
```

Then you can just run the commands for the steps below by typing `raxml` at the command line. If you get an error that the command is not found, make sure the directory `~/local/bin` is included in your `$PATH` environment variable (which is where UNIX searches for tools):

```
export PATH=$PATH:$HOME/local/bin
```

3.2 Run RaxML

Now, use RaxML to build a tree. We will first build a maximum likelihood tree, and annotate the branches with confidence values, as was done in the NEJM figure. Even though they didn't give us much detail, we do know:

- They used bootstrapping to annotate support for each split. Their methods say they did 1000 bootstraps. We'll do fewer to save time. Our solution used 100. But you might want to try with fewer to make sure things are working first.
- They used a "general time-reversible model". (Hint look for "GTRCAT" in the RaxML manual).

We'll actually need to use multiple RaxML commands:

- The first will find the maximum likelihood tree based on our mafft alignment.
- The second will perform the bootstrap search. This one can take a while. You might want to use `nohup`.
- The third will draw the bipartitions (bootstrap values) on the best tree generated by the first command. This should be fast.

Figure out from the RaxML manual, or from googling, how to do this. You might find Step 4 of this tutorial helpful: https://cme.h-its.org/exelixis/web/software/raxml/hands_on.html.

You should end up with a file named something like `RAxML_bipartitionsBranchLabels.lab6_raxml_bs` with your final tree containing branch labels with the bootstrap values. This file is in newick format, a common file format to describe trees.

4. Visualize the tree

Finally, visualize your tree. We recommend you use one of multiple online tools for viewing trees from Newick files. e.g.:

- iTOL
- ETE Treeview

For both of these, you can directly copy the text of your newick file into the text box provided and visualize the output. You might want to improve on the intitial visualization they provide. e.g.:

- Make sure bootstrap support values are displayed at the branchpoints.
- You may want to change the node labels, either programmatically or manually, to make interpreting your tree easier. For instance, rather than display accessing codes like AY508724.1 you could label the nodes something like "SARS coronavirus NS-1" based on the annotations here: https://docs.google.com/spreadsheets/d/1p1JpKKj1lUmGqrq2fdnX-FHvv7wt-jIdyY4qrRXyT9I/edit?usp=sharing.
- You'll want to at least be able to pick out which leaves correspond to COVID-19 samples, the original SARS, and MERS.

5. Summarize your findings

In the cell below:

- Briefly summarize the methods you used
- Include your tree
- Summarize your main findings

Remember to document your code in the README.

Question 1:

I first started with downloading the data by using the wget command with the link provided that contained the KC545386 sequence in fasta format and used the awk command to go through each line of the accession txt file and KC545386 was replace for each accession number. The command then added the fasta sequence for each accession number to a fasta file. Then, I ran another code which removed all the empty lines in the fasta file and outputted to lab7_virus_genomes_fa.

Question 2:

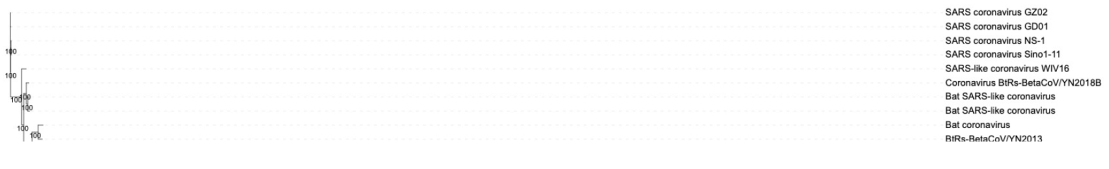
I then installed mafft using git clone and then used vi to edit the Makefile's top line. I then saved the Makefile and installed mafft using these commands in the following order make clean, make, make install. To run mafft I used mafft in the command line which opened the interactive menu. I used lab7_virus_genomes.fa as the input file and chose Output format as 5. Phylip format / Sorted and used the default strategy of 3. FFT-NS-2 (default). I then had the output file as lab7_virus_genomes.aln.

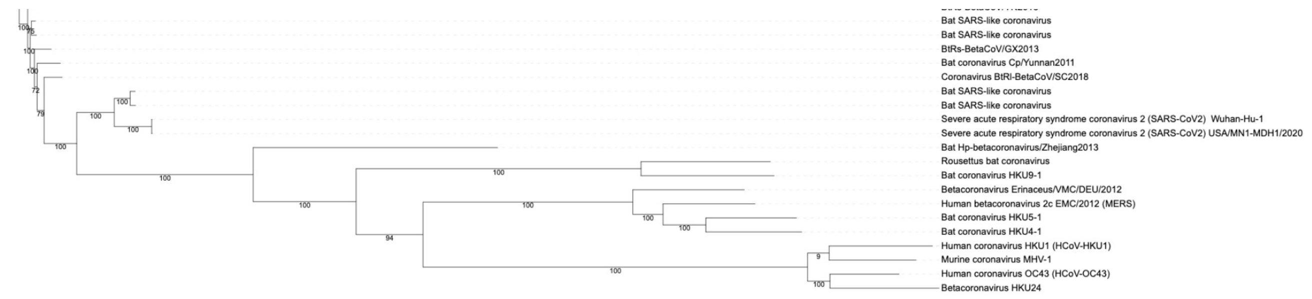
Question 3:

Then I installed RAxML by downloading the .tar.gz file for version 8.2.12. I then ran gunzip on the .tar.gz file to unzip it and then ran tar xf on the .tar file. I then used the make -f command on Makefile.gcc, Makefile.SSE3.gcc, Makefile.PTHREADS.gcc, Makefilee.SSE3.PTHREADS.gcc and ran rm to remove all the .o files after each make command except for SSE3.PTHREADS.gcc the last time. This returned a file called raxm1HPC-PTHREADS-SSE3 which I copied to ~/local/bin/raxml directory which is the local path. I tried running raxmlHPC and raxml command but it said command not found for both so I then ran the export command to set path to the local path. Then, to run RAxML I first did the simple ML search used the raxml command and used the lab7_virus_genomes.aln file as the input with the method GTRGAMMA and -p option of 12345 which was outputted to .T1 files. I then ran the bootstrapping search using the raxml command again with the same commands except I added a -b option of 12345 too and I changed it to output to .T2 files. Lastly, I ran the biparitions search using raxml command once again but this time I used the method of GTRCAT in order to use the "general time-reversible model". I used the same -p option value but this time I added -f option of b and -t option to the RAxML_bestTree.T1 file and used -z option for RAxML_bootstrap.T2 file and outputted it to .T3 files.

Question 4:

For Question 4 I used the iTOL website and uploaded the RAxML_bipartitionsBranchLabels.T3 file. I then went through each label and replaced it with annotation name. I also displayed the bootstrap support values. I noticed that the first couple leaves are for the SARS GZ0Z, GD01, Sino1-11, and NS1 coronavirus samples. The next breaks in the tree are Coronavirus BtRs-BetaCoV/YN2018B sample and a mixup of mainly most of the Bat SARS-like coronavirus and Bat coronavirus samples. Then, there are two leaves of both the Severe acute respiratory syndrome coronavirus 2 samples. The next few leaves are the reminder of the bat coronavirus samples which include Bat Hp-betacoronavirus/Zhejiang2013, Rousettus bat coronavirus, Bat coronavirus HKU9-1, Bat coronavirus HKU5-1, and Bat coronavirus HKU4-1 samples. The last four leaves are the Human coronavirus HKU1 (HCoV-HKU1), Murine coronavirus MHV-1, Human coronavirus OC43 (HCoV-OC43), and Betacoronavirus HKU24 samples. Below is the tree I made and visualized using the RAxML_bipartitionsBranchLabels.T3 file.


100
100
100
100
100
100
100
SARS coronavirus GZ02
SARS coronavirus GD01
SARS coronavirus NS-1
SARS coronavirus Sino1-11
SARS-like coronavirus WIV16
Coronavirus BtRs-BetaCoV/YN2018B
Bat SARS-like coronavirus
Bat SARS-like coronavirus
Bat coronavirus
BtRs-BetaCoV/YN2013



Bat SARS-like coronavirus
Bat SARS-like coronavirus
BtRs-BetaCoV/GX2013
Bat coronavirus Cp/Yunnan2011
Coronavirus BtRl-BetaCoV/SC2018
Bat SARS-like coronavirus
Bat SARS-like coronavirus
Severe acute respiratory syndrome coronavirus 2 (SARS-CoV2) Wuhan-Hu-1
Severe acute respiratory syndrome coronavirus 2 (SARS-CoV2) USA/MN1-MDH1/2020
Bat Hp-betacoronavirus/Zhejiang2013
Rousettus bat coronavirus
Bat coronavirus HKU9-1
Betacoronavirus Erinaceus/VMC/DEU/2012
Human betacoronavirus 2c EMC/2012 (MERS)
Bat coronavirus HKU5-1
Bat coronavirus HKU4-1
Human coronavirus HKU1 (HCoV-HKU1)
Murine coronavirus MHV-1
Human coronavirus OC43 (HCoV-OC43)
Betacoronavirus HKU24
100
100
100
72
79
100
100
100
100
100
100
100
94
100
100
100
100
9
100

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js