**BIOMI 609 Computational Genomics and Bioinformatics**

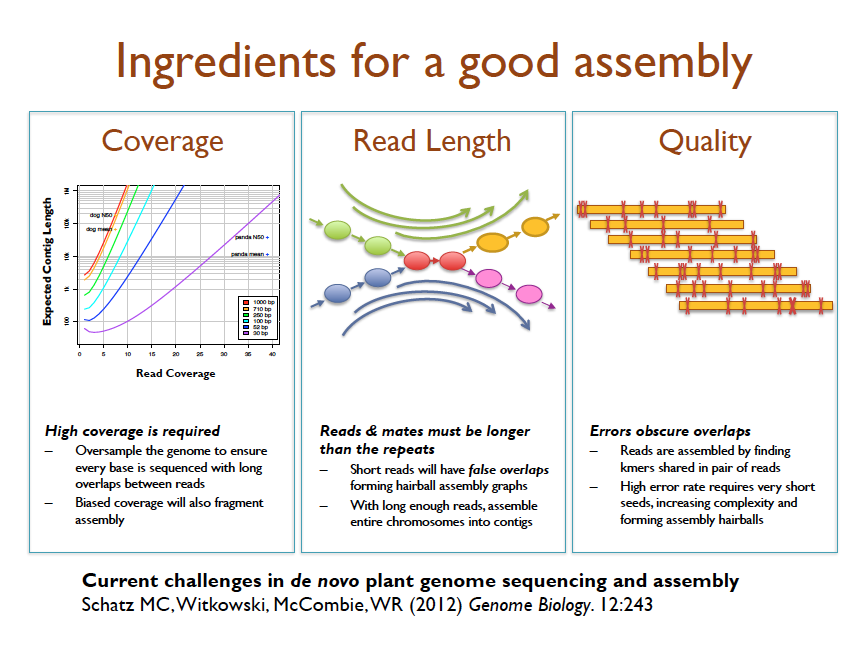
**Spring 2022**

**Lab 1 NGS Data Analyses – QC, Assembly**

**Dr. Arun Sethuraman**

Starting this week, and proceeding through the next few weeks of the semester, we will learn to work with short-read genomic data, popularly generated through “next-generation sequencing” technologies. This week, we will focus on the first step of NGS data analyses, which is genomic assembly.

Genome assembly is a difficult problem in a lot of aspects – (1) biological – species may have varying ploidy levels, heterozygosity may be hard to resolve (recall that humans for instance are diploid, with a maternal and a paternal copy of each genome), genomes may contain varying levels of repeat content, (2) sequencing – genomes can be large, and imperfect sequencing (garbage-in-garbage-out) can lead to erroneous downstream analyses, (3) computational – genomes can be large and complex, making them that much harder to assemble, (4) accuracy – it is also very difficult to assess the correctness of a genome’s assembly.



We will focus on SARS-CoV2 genomes that are publicly available via [www.gisaid.org](http://www.gisaid.org) and the Sequence Read Archive (<https://www.ncbi.nlm.nih.gov/sra>). For today’s lab, we will (1) learn how to login and use our XSEDE accounts via Jetstream, (2) download and assess quality distribution of reads from a single SARS-CoV2 genome, (3) assemble this genome *de novo* using a popular software for genome assembly using short read data, called Velvet, and (4) assess the quality of the obtained assembly.

**Exercise 0 – Logging onto your XSEDE Virtual Machine via Jetstream**

Details on Jetstream: <https://portal.xsede.org/jetstream>

First things first, please login to your XSEDE account, and set up DUO authentication. Instructions on this can be found here: <https://portal.xsede.org/mfa>

Once you’ve done this, log in through this website: <https://use.jetstream-cloud.org/> with your XSEDE username and password; approve it on DUO. This should take a few seconds, then open Jetstream in your browser.

Now you need to launch a VM instance for you to use – click on “Launch New Instance”, and this will take you to an Image Search window. Scroll down/search for “Genomics Toolkit” and click on it, then hit “Lauch”. If it asks for resources, state “m1.medium”, which contains 6 CPU’s, 16 GB of memory, and disk space of 60 Gb. This should hopefully suffice for our purposes of assembling, aligning, etc. SARS-CoV2 genomes.

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

This will take a few minutes (up to 30 minutes), after which the screen should update itself to “Active”, with a green button by it (or at the very least, you should get an email stating that the machine has been launched and should show up as active in the “Instance Status History”. This means that your VM has been launched successfully. Thereon, to access your VM, click on “Open Web Shell”, or alternately you can also click on “Web Desktop” and this should launch the VM into a new browser tab. Now just open a Terminal inside your VM.

**Exercise 1 – Installing Velvet and SRA Toolkit**

Prior to any installations, make sure to create a folder for yourself and your work in your home folder. Once you’ve launched a terminal, typing pwd will tell you your current working directory. We will review some common Unix commands first.

Read the velvet manual: <https://www.ebi.ac.uk/~zerbino/velvet/Manual.pdf>

Thereon, create a new directory in /home/username/, call it Tools, then cd into it:

mkdir Tools

cd Tools

To download and install velvet, type:

wget <https://www.ebi.ac.uk/~zerbino/velvet/velvet_1.2.10.tgz>

tar -zxvf velvet\_1.2.10.tgz

cd velvet\_1.2.10

make ’OPENMP=1’

#This should install velvet into your current folder.

Let’s set the PATH to this folder, so it’s made accessible anywhere in your terminal.

pwd

This will print out the current working directory, which should look something like /home/username/velvet\_1.2.10

Now to add this to the PATH:

export PATH=$PATH:/home/username/velvet\_1.2.10

Now cd out of it:

cd ..

#To install SRA Toolkit

wget <https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/2.11.3/sratoolkit.2.11.3-centos_linux64.tar.gz>

tar -zxvf sratoolkit.2.11.3-centos\_linux64.tar.gz

vdb-config --interactive

#Then just save and exit (press s, then x).

#This automatically creates the needed tools inside the sratoolkit.2.11.2-centos\_linux64/bin folder

Now remember that we need to set the PATH for all applications to be accessible. So let’s do that. cd into the bin folder, thereon type pwd, and copy the path printed.

Then set PATH as:

export PATH=$PATH:/home/username/sratoolkit.2.11.2-centos\_linux64/bin

VOILA! Now cd out of this folder, create another folder, call it Week1, and cd into it.

**Exercise – 2**: **Downloading SARS-CoV2 data, processing it with SRA Toolkit, fastqc, trimmomatic**

First we need to download an SRA formatted set of data for a genome of your choice – so go to Sequence Read Archive, and click on ‘SARS-CoV-2 data (NCBI)’, which should take you to this page: <https://www.ncbi.nlm.nih.gov/sars-cov-2/>

Now scroll down to “…next-generation sequencing runs in SRA”, and click on “View in SRA”. I then filtered for “genomes”, and specifically “paired” end sequencing reads.

Graphical user interface, text, application, email

Description automatically generated

Pick one – here’s one:

Graphical user interface, text, application, email

Description automatically generated

There’s a lot of information about the platform, # of bases sequenced, its size, etc. shown. If you click on the particular Run, it should take you to an SRA page where you can download the data if you click on “Data access” (see the sra-download link that is printed). Just copy the link, then you can download it inside your terminal window on your VM.

wget <https://sra-download.ncbi.nlm.nih.gov/traces/sra53/SRR/017435/SRR17854410>

This should download the SRA for this genome onto your folder. Now to convert this to FASTQ, we need to use the fastq-dump tool in SRA Toolkit.

fastq-dump -I --split-files SRR17854410

This should create a pair of FASTQ files in your folder (confirm by typing ls, head, etc).

Now let’s assess quality of these reads!

fastqc SRR17854410\_1.fastq SRR17854410\_2.fastq

This automatically runs fastqc on your files, and generates reports, which should be stored as HTML files in your folder. You can view these by clicking on Applications->File Manager, then go to your Week1 folder, double click on the HTML file. Open these files, and explore the QC distribution, look for adapter contamination, and other points of concern.

**Extra:** Use trimmomatic to trim these reads based on Q score distributions - use a Q score cutoff of 30. Manual for trimmomatic can be accessed here: <https://datacarpentry.org/wrangling-genomics/03-trimming/index.html>

For example, in my case, the fastqc plot appears like the last few trailing bases seem to be < Q score of 30. So if I want to just go ahead and trim everything below a 30 Q score, I can use this command:

trimmomatic PE -threads 6 SRR17854410\_1.fastq SRR17854410\_2.fastq SRR17854410\_1\_trimmed.fastq SRR17854410un\_1.fastq SRR17854410\_2\_trimmed.fastq SRR17854410un\_2.fastq TRAILING:30

**Exercise – 3 Assembly using VELVET**

(Adapted from EBI tutorial: <https://www.ebi.ac.uk/training/online/sites/ebi.ac.uk.training.online/files/user/18/private/velvet-practical_part-1.pdf> and <https://www.ebi.ac.uk/training/online/sites/ebi.ac.uk.training.online/files/user/18/private/velvet-practical_part-2.pdf>):

1) Time to run velveth and velvetg. Couple of important UNIX commands to be introduced here:

It would be interesting to monitor the way that the programs you will run utilize your computer's resources, particularly memory. A simple way to do this is to open a second terminal and in it type top.

top is a program that continually monitors all the processes running on your computer, showing the resources used by each. Alternatively you could use a System monitor from your operating system as well. Leave this running and refer to it at intervals, especially when programs appear to be taking a long time, of just whenever your curiosity gets the better of you. You should find that as this practical progresses, memory usage will increase significantly, but hopefully not beyond the capacity of your workstation.

Now, back to the first terminal, you are ready to run velveth and velvetg. The reads are -shortPaired this time and for the first run you should not use any parameters for velvetg. From this point on, where it will be informative, time your runs. This is very easy to do, just prefix the command to run the program with the command time. This will cause UNIX to report how long the program took to complete its task. Set the two stages of velvet running, whilst you watch the memory usage as reported by top. time the velvetg stage. The commands to enter are:

Let’s start with a De Bruijin graph k-mer length of 25, and an output directory called run\_25

time velveth run\_25 25 ­-shortPaired -separate -­fastq SRR17854410\_1\_trimmed.fastq SRR17854410\_2\_trimmed.fastq

velveth will do this run for a bit, then output some files to the directory. Move into this folder and look around. HINT: You can use the command less in UNIX to look at files, press q to quit.

What do you find? Describe the contents of the two velveth output files. What does the Log file store?

Thereon, move out of this directory, and run velvetg

time velvetg run\_25

Now move back into the run\_25 folder and observe what velvetg did. What extra files do you see in the folder? What do they represent? In the Log file in run\_25, what is the N50?

Hopefully, we will have discussed what the N50 statistic is by this point. Broadly, it is the median (not average) of a sorted data set using the length of a set of sequences. Usually it is the length of the contig whose length, when added to the length of all longer contigs, makes a total greater that half the sum of the lengths of all contigs. Easy, but messy – a more formal definition can be found here: <https://www.molecularecologist.com/2017/03/whats-n50/>

In order to improve our results, take a closer look at the standard options of velvetg by typing

'velvetg' without parameters. For the moment focus on the two options -cov\_cutoff and -exp\_cov.

Clearly -cov\_cutoff will allow you to exclude contigs for which the kmer coverage is low,

implying unacceptably poor quality. The -exp\_cov switch is used to give velvetg an idea of the coverage to expect. If the expected coverage of any contig is substantially in excess of the suggested expected value, maybe this would indicate a repeat. For further details of how to choose the parameters, go to 'Choice of a coverage cutoff': <https://github.com/dzerbino/velvet/wiki/Manual>

Also, one way of obtaining the expected coverage is by determining using the formula C = LN/G, where C = coverage, G = length of the genome, L = read length, N = # of reads. So for the example I’m using, G = 29800 bp, N = 12478, L = 150 bp, so C = 62.8, or in other words, we would expect each nucleotide location to be present on an average across ~63 reads.

Next, after saving your contigs.fa file from being overwritten, set the cut-off parameters that you investigated in the previous exercise and rerun velvetg. time and monitor the use of resources as previously. Start with -cov\_cutoff 16 thus:

mv run\_25/contigs.fa run\_25/contigs.fa.0

time velvetg run\_25 -cov\_cutoff 16

Save your contigs.fa file again and try velvetg with both -cov\_cutoff 16 and -exp\_cov 63. Until this run, velvetg ignored the paired-end information. By using -cov\_cutoff and -exp\_cov, velvetg tries to estimate the insert length, which you will see in the velvetg output. The command is, of course:

mv run\_25/contigs.fa run\_25/contigs.fa.1

time velvetg run\_25 -cov\_cutoff 16 -exp\_cov 63

How much time is required for all the above velvetg runs? Which insert length does velvet estimate?

Try giving the -cov\_cutoff and/or -exp\_cov parameters the value auto – the velvetg help output could show you how. The information velvet prints during running includes information about the values used (coverage cut-off or insert length) when using the “auto” option.

Which coverage values does velvet end up choosing? (HINT: Look at the output which velvet produces while running) How does the N50 value change?

**Exercise 4 - Assembly quality assessment using QUAST**

Now go to your Tools folder and install QUAST, which is a genome quality assessment tool. Details about QUAST can be found here: <https://github.com/ablab/quast>

To install, go to your Tools folder, then type:

git clone <https://github.com/ablab/quast.git>

cd quast

./install.sh

This should install QUAST in your current folder. Set the path to this folder.

Now we need a SARS-CoV2 reference genome for us to compare our assembly to. The reference genome can be accessed via the SRA. To do this, launch a browser window inside your Atmosphere client, and search for SARS-CoV2 reference genome; download the genome as a FASTA (I’ll show you how to do this), also download the genome annotation as a GFF. Move these into your Week 1 folder. Then run quast as:

quast.py run\_25/contigs.fa run\_25/contigs.fa.0 -r reference.fasta -g GCf\*.gz

Now view all the result files that are printed in the quast\_results folder.

**Extra - other (and better) assemblers:**

Velvet is just a quick and dirty solution - and it’s definitely NOT the best *de novo* assembler out there. You can compare a few different assemblers. Here are a few that you can install on your VM, and play with:

1) ABySS - <https://github.com/bcgsc/abyss>

2) SOAPdenovo2 - <https://github.com/aquaskyline/SOAPdenovo2>

3) Unicycler - <https://github.com/rrwick/Unicycler>

4) SPAdes - <http://cab.spbu.ru/software/spades/>