Ecological Genomics Tutorials: Population & Landscape Genomics 2

September 13, 2023

Learning Objectives for 09/13/23

1. Review visualizing QC on our sequence reads
2. Trim reads using `fastp` and assess pre- and post trimming
3. Start mapping (a.k.a. aligning) each set of cleaned reads to a reference genome
4. Visualize sequence alignment files
5. Calculate mapping statistics to assess quality of the result

Clean using `fastp`

Now that we have a sense of the quality of our data, and where we might need some trimming, our next step is to clean the reads by removing low quality bases and leftover Illumina sequence adapters from the sequences.

We'll use the `fastp` program to clean the reads for each file. See also the paper published by Chen in May 2023. Another very similar program for trimming is Trimmomatic. It works great, but `fastp` does everything Trimmomatic does but in only a faction of the time! The program is already installed on our server:

`/data/popgen/fastp`

This job will be bit more complicated, since we want you each to process all the samples for a given population.

Since we're doing multiple samples from the same population, we want to be clever and process in a batch instead of manually one at a time. We can do this by writing a bash script that contains a loop. This is the first taste of how bash scripting is powerful!

The basic syntax of a bash loop is like this:

```
cd <path to the input data>

for FILE in somelist

do

  command1 -options ${FILE} -moreOptions
  command2 -options ${FILE} -moreOptions

done
```


Note the use of variable assignment using ${}. We define the word `FILE` in the "for loop" as the variable of interest, and then call the iterations of it using ${FILE}. For example, we could use the wildcard character (*) in a loop to call all files that include the ID code for your population and then pass those filenames in a loop to the commands. For example, if your pop was "2XXX", then you could write a loop that would process all the R1 fastq files in that population like this:

```
MYPOP="2XXX"

cd <path to the input data>

for FILE in ${MYPOP}*R1.fastq.gz

do

  command1 -options ${FILE} -moreOptions
  command2 -options ${FILE} -moreOptions

done
```

That's the basic idea! Let's do it...

We've provided a partially completed example script here:

`/netfiles/ecogen/PopulationGenomics/scripts/fastp.sh`

1. Make a copy of the script `cp` and put it into your `~/myscripts` directory
2. Open and edit your copied script using the program vim `vim ~/myscripts/fastp.sh`.
3. Edit the file so that you're trimming the fastq files for the population code assigned to you; you can add annotations as notes to yourself using the hashtag (#)
4. Save the file after you're done making changes. To do this, hit to get out of "Insert" mode, then `:wq` This will write (w) the changes to your file and then quit (q) vim.

NOTE: `fastp` needs both read pairs (i.e., the R1 and R2 files) given to it at the same time. This makes the use of name variables in the loop a bit tougher since we can't rely just on the population name (example, `2505`) but also need each individual name too (example, `2505_9_C`).

We'll use a couple of tricks to recycle the R1 file name to quickly create a mathcing R2 file name within the loop. We'll do a simialr trick to quickly name the output files adding "_clean" to the end. We'll talk about this in class together, but there are also annotations in the script itself explaining what's happening.

Once you're ready, go ahead and execute your bash script that you saved into your `~/myscripts/` folder:

```
cd ~/myscripts/
bash fastp.sh
```

Note that the output (the trimmed and cleaned reads) are going to get saved to a folder on the network drive that you've been given write access to:
`/netfiles/ecogen/PopulationGenomics/fastq_red_spruce/cleanreads`

Visualize pre- and post-trimming read quality with the `fastp` html output

Conveniently, `fastp` is not only fast, but also produces a summary of the change in quality pre- and post-trimming. Just like FastQC, the output is in an html file that we can use FileZilla to transfer back to our laptops and look at in a browser. Pretty cool!

Mapping cleaned and trimmed reads against the reference genome

Now that we have cleaned and trimmed read pairs, we're ready to map them against the reference genome.

The first step of read mapping is downloading the reference genome, freely available from the developers at plantgenie.org:

The *Picea abies* reference genome is based on Norway spruce (*P. abies*) and published by Nystedt et al. (2013).

Genome	
Size (1*n*)	19.6 Gb
Karyotype	$2n = 24$
GC content	37.9%
High-copy repeat content*	
LTR_*Gypsy*/*Copia*/unclassified	35%/16%/7%
LINE	1%
DNA transposable element	1%
Unclassified	10%
Genes and gene-like fragments†	2.4%
Assembly	
Size in scaffolds >200 bp/>10 kb	12 Gb/4.3 Gb
N50/NG50	4,869 bp/721 bp
Annotation	
High confidence gene set	28,354
Genes with >5-kb introns	8.4%
Avg. exon/intron size	312 bp/1,017 bp
Avg. gene density	1 gene in 705 kb
Transposable element genes	284,587
Non-coding loci	
lncRNA (unique/conserved)	13,031/9,686
miRNA (*de novo* predicted)	2,719

* Inferred from unassembled reads. † Including pseudogenes, excluding transposable elements.

You don't actually need to download the genome because we already have the file in the directory below. But for future reference `wget` is a useful command to ownload files from the web.

```
cd /netfiles/ecogen/PopulationGenomics/ref_genome
wget "ftp://plantgenie.org:980/Data/PlantGenIE/Picea_abies/v1.0/fasta/GenomeA
```

Rather than trying to map to the entire 19.6 Gbp reference (yikes!), we first subsetted the *P. abies* reference to include **just the contigs that contain one or more probes** from our exome capture experiment. For this, we did a BLAST search of each probe against the *P. abies* reference genome, and then retained all scaffolds that had a best hit.

- This reduced reference contains:
  - 1,376,182,454 bp (~1.38 Gbp) in 33,679 contigs
  - The mean (median) contig size is 10.5 (12.9) kbp
  - The N50 of the reduced reference is 101,375 bp
- The indexed reduced reference genome to use for your mapping is on our server here:

`/netfiles/ecogen/PopulationGenomics/ref_genome/Pabies1.0-genome_reduced.fa`

To help make our scripting approach efficient, we're going to write several short scripts, optimizing each one at a time, then put them together at the end

- First, we want to specify the population of interest and the paths to the input and output directories. We can do this by defining variables in bash, like so:
- Each student gets assigned a population to work with:
- `MYPOP="XXXX"`
- Directory with the cleaned fastq files
- `INPUT="/netfiles/ecogen/PopulationGenomics/fastq/red_spruce/cleanreads"`
- Output dir to store mapping files (bam)
- `OUT="/netfiles/ecogen/PopulationGenomics/fastq/red_spruce/cleanreads/bam"`
- For mapping, we'll use the program bwa, which is a very efficient and very well vetted read mapper. Lots of others exist and can be useful to explore for future datasets. We tried several, and for our exome data, bwa seems to be the best.
- Let's write a bash script called `mapping.sh` that calls the R1 and R2 reads for each individual in your population, and uses the `bwa-mem2` algorithm to map reads to the reference genome. We can test this out using one sample (individual) at a time, and then once the syntax is good and the bugs all worked out, we can scale this up to all the inds in our populations.

The basic `bwa-mem2` command we'll use is below. Think about how we should write this into a loop to call all the fastq files for our population of interest...(hint, look back at the `fastp.sh` script)

```
/data/popgen/bwa-mem2/bwa-mem2 mem -t 1 ${REF} ${READ1} ${READ2} > ${OUT}${NA
```

where

```
-t 1 is the number of threads, or computer cpus to use (in this case, just 1)
-${REF} specifies the path and filename for the reference genome
${READ1} specifies the path and filename for the cleaned and trimmed R1 reads
${READ2} specifies the path and filename for the cleaned and trimmed R2 reads
>${OUT}/${NAME}.sam  specifies the path and filenam for the .sam file to be s
```

- Other bwa options detailed here: bwa manual page

Because you're each mapping sequences from multiple samples (N=8/pop), it's going to take a little while.

Whenever you have a job that will take a long time, you're going to want to start a "screen session" using the `tmux` command. `tmux` initiates a new shell window that won't interrupt

or stop your work if you close your computer, log off the server, or leave the UVM network. Anytime you're running long jobs, you definitely want to use `tmux`.

Using it is easy. Here's what you need to do:

- Type `tmux` and hit `<return>`
- You are now in a "screen" — meaning you can start a job and once you exit the screen it will keep running.
- Start your program or run your bash script like so: `bash mapping.sh`
- You'll see the program start running. - Assuming everything is working and you're not seeing errors, you want to now detach from the screen. To do so, hold the `<control>` key down while you hit the `b` key. Then release the `<control>` key and hit just the `d` key.
- If you do it right, then you should exit your screen session and go back to a "regular" terminal. If you don't do this, you'll lose your work!
- You can recover your screen by typing `tmux attach`. That'll re-attach you back to your program!

When your script is ready, do:

```
tmux

bash mapping.sh
```

Don't forget to detach from your screen!

While that's running, let's take a look at a Sequence AlignMent (SAM) file already available in `/netfiles/ecogen/PopulationGenomics/fastq/red_spruce/cleanreads/bam/`

- First, try looking at a SAM file using `head` and `tail`.

```
tail -n 100 FILENAME.sam
```

A SAM file is a tab delimited text file that stores information about the alignment of reads in a FASTQ file to a reference genome or transcriptome. For each read in a FASTQ file, there's a line in the SAM file that includes

- the read, aka. query, name,
- a FLAG (number with information about mapping success and orientation and whether the read is the left or right read),
- the reference sequence name to which the read mapped
- the leftmost position in the reference where the read mapped

- the mapping quality (Phred-scaled)
- a CIGAR string that gives alignment information (how many bases Match (M), where there's an Insertion (I) or Deletion (D))
- an '=', mate position, inferred insert size (columns 7,8,9),
- the query sequence and Phred-scaled quality from the FASTQ file (columns 10 and 11),
- then Lots of good information in TAGS at the end, if the read mapped, including whether it is a unique read (XT:A:U), the number of best hits (X0:i:1), the number of suboptimal hits (X1:i:0).

The left (R1) and right (R2) reads alternate through the file. SAM files usually have a header section with general information where each line starts with the '@' symbol. SAM and BAM files contain the same information; SAM is human readable and BAM is in binary code and therefore has a smaller file size.

Find the official Sequence AlignMent file documentation can be found here or more officially.

- Some useful FLAGs to know - for example what do the numbers in the second column of data mean?

- Here's a SAM FLAG decoder by the Broad Institute.

How can we get a summary of how well our reads mapped to the reference?

- We can use the program sambamba for manipulating sam/bam files. sambamba. Sambamba is closely related to its progenitor program `samtools` which is written by the same scientist who develop `bwa`, Heng Li. `sambamba` has been re-coded to increase efficiency (speed).

- First we have to convert the sam file to bam format:

- `sambamba view -S -f bam IN.sam -o OUT.bam`

- Then we can use the command `flagstats` gets us some basic info on how well the mapping worked:

- `sambamba flagstat FILENAME.bam`

Let's talk about how our mapping went!