Small Group Exercise #2

Contig Assembly with Trinity and Ray

Part 1 --- Trinity

1. Return to the ASC documentation system https://hpcdocs.asc.edu/ and explore the README info for the transcriptome assembler **Trinity**. This file includes various problems the system administrators have noticed and multiple examples of how to run the software.

    i. Go to https://hpcdocs.asc.edu/ and log in with your ASC credentials.
    ii. Click on "Sequence_assembly", then "Trinity".

2. From the README info above, you will need a script to run Trinity. A template named `Trinity_example.sh` is included in the shared example scripts folder.

3. You will need to make a copy of the above script to your home directory. Let's do that:

    i. `cd` (to take you to the top level of your home directory)
    ii. `mkdir trinity_working_example` (create a directory to work in)
    iii. `cd trinity_working_example` (to change into the directory)
    iv. `cp /home/shared/biobootcamp/data/example_ASC_queue_scripts/Trinity_example.sh .` (note the period at the end of the command, which means "copy the script to your current location, using the same filename)

4. Now use either a text editor, `less`, or `cat` to examine the script:
a. e.g. `nano Trinity_example.sh`. Read through the entire script to get an overall idea of what is being done.

5. Note that comments in the script indicate where to edit once you decide exactly what you want to do. Also note that this script is much more complex than what you saw with the `FASTX_example.sh` script. The reason for this can be found in the Trinity README text (under ## ADVANCED EXAMPLE) as well as in the comments embedded in the script. Close nano by typing `Ctrl-x` on your keyboard.

6. Let's copy some raw FASTQ data to our working directory. **Here, and throughout the workshop, use the convenience of tab-completion and command-recall (up arrow) to save time and avoid typos**.
`cp /home/shared/biobootcamp/data/Lamellibrachia_luymesi_sequence_reads_for_assembly/Lamellibrachia_luymesi_transcriptomic_sub1M_L001_R1_001.fastq .`

`cp /home/shared/biobootcamp/data/Lamellibrachia_luymesi_sequence_reads_for_assembly/Lamellibrachia_luymesi_transcriptomic_sub1M_L001_R2_001.fastq .`

7. Quantify the number of reads in each file and error check them using `wc -l` and fastQValidator (as discussed in lecture). NOTE: While you worked with these files earlier, the fact that you just created new copies of them introduces a potential source of error e.g. a truncated or corrupted copy. Remember: **always employ controls**.

8. Open the `Trinity_example.sh` script with nano and add the names of the `*.fastq` files to their appropriate sections. NOTE: the `*_L001_R1_001.fastq` file should be associated with the `--left` option while the `*_L001_R2_001.fastq` file is associated with the `--right` option. These R1 / R2 names are standard for Illumina paired-end reads.

9. Save your script and exit out of nano. Submit it to the ASC queue system **using the directions at the bottom of the script**.

10. Check the status of your job using `squeue` [try squeue - tab complete to see additional queue commands]

While the transcriptome assembly is "baking", proceed to the genomic assembly...

Part 2 --- Ray

1. Explore the documentation for the genome assembler **Ray** on the ASC website:

    i. Go to https://hpcdocs.asc.edu/ and log in with your ASC credentials.
    ii. Click on "Sequence_assembly", then "Ray".

2. From the README above, you will need a script to run Ray. An example script named `RAY_example.sh` is located at:
`/home/shared/biobootcamp/data/example_ASC_queue_scripts`

3. You will want to make a copy of the script to your current Ray workspace (directory). Let's do that:

    i. `cd` (to take you – wherever you are – to your home directory).
    ii. `mkdir ray_working_example` (create a new directory to work in).
    iii. `cd ray_working_example` (to move into the new directory).
    iv. `cp /home/shared/biobootcamp/data/example_ASC_queue_scripts/RAY_example.sh .` (to copy the script to your current location).

4. Now use a text editor to open the script: e.g. `nano Ray_example.sh`

5. Note again that comments in the script indicate where to edit once you determine what you want to do. Close nano by pressing “Ctrl” and then “x” together on your keyboard (hereafter abbreviated `Ctrl-X`.

6. Let’s copy some different raw FASTQ files to our working directory:
   `cp /home/shared/biobootcamp/data/Lamellibrachia_luymesi_sequence_reads_for_assembly/Lamellibrachia_luymesi_genomic_L001_R1_001.fastq .`
   `cp /home/shared/biobootcamp/data/Lamellibrachia_luymesi_sequence_reads_for_assembly/Lamellibrachia_luymesi_genomic_L001_R2_001.fastq .`
   hint: use up arrow to recall previous command and then edit it inline using left and right arrow keys, `Ctrl-A` and `Ctrl-E`

7. Quantify the number of reads in each file and error check them using `wc -l` and fastQValidator (as done previously) and make note of what they are.

8. Open the `Ray_example.sh` script with nano and add the names of the `*.fastq` files to their appropriate sections. NOTE: the `*_L001_R1_001.fastq` file should precede the `*_L001_R2_001.fastq` file going left to right following the `-p` option. Also set the `-o` option to an appropriate name of your choosing.

9. **NOTE:** for this part of the exercise, your group needs to discuss the first option, namely potential values for `-k` (which sets the k-mer length used in the assembly). Base your discussion on the materials covered during the lecture. Some specifics regarding values for k: a) the default value is 21; b) the value must be odd. The odd k-mer size preserves the direction of repetitive or palindromic sequences when they are reverse complemented during the assembly process. For example forward strand ATAT equals its reverse complement ATAT, but forward strand ATATA does not equal its reverse complement TATAT. This simple trick helps the assembler put the kmers together more efficiently and with less ambiguity; c) larger k-mers utilize more memory. Once the group has selected four potential values for the `-k` option, each person will add one of these values to their `Ray_example.sh`, replacing (hint: or using) the value of –k 31 that is already there. Be forewarned: extreme outliers for `-k` might lead to the assembly imploding (and taking you with it).

10. Save your script and exit out of nano. Now submit it to the ASC queue system using the directions at the bottom of the script. Hint: `cat` it out to have in front of you while you enter parameters into the queue.

11. Monitor this job (as well as your Trinity run) using `squeue`

Notes:

• Everyone has a symlink or shortcut to the shared bootcamp directory called `biobootcamp` in their home folder. You can type “biobootcamp” in place of the absolute path `/home/shared/biobootcamp/` e.g. `cat biobootcamp/data/example_ASC_queue_scripts/Trinity_example.sh` for this exercise and all other bootcamp activities this week. If you aren’t currently in your home directory, we use `~/biobootcamp` , where the tilde always means *your* home directory.

• Resize your terminal window and choose font and screen background color to suit your preference and increase legibility.

• Try the command `echo $$` by itself. This number is unique to your login session and gives your file variables a unique name as a result. Lookup the man page for the command `ps`, then try typing `ps` from the prompt. Recognize your number?