Small Group Exercise #3

Contig Assessment with BLAST+

Part 1 --- Characterize results from a genomic assembly

1. Return to the ASC documentation system https://hpcdocs.asc.edu/ and explore the README info for the command line version of BLAST+

   a. Go to https://hpcdocs.asc.edu/ and log in with your ASC credentials.
   b. Click on "Sequence analysis", then "Blast+".

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

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

   i. `cd` (to take you to the top level of your home directory)
   ii. `mkdir blast+_working_example` (create a directory to work in)
   iii. `cd blast+_working_example` (to change into the directory)
   iv. `cp /home/shared/biobootcamp/data/example_ASC_queue_scripts/BLAST+_example.sh .` (to copy the script to your working directory)

4. Now use your choice of text editor to open the script. For example to use nano: `nano BLAST+_example.sh`

5. Note that comments in the script indicate where you will be tweaking your command once you decide exactly what you want to do. Close nano with `Ctrl-x`.

6. Now make a copy of the contig file resulting from **your** Ray genomic assembly performed previously to the `blast+_working_example` directory. The file should be named `Contigs.fasta` and be located in the output directory **you** specified in the `Ray_example.sh` script within your `ray_working_example directory` (Ex. 2). If for some reason you need a copy of a Ray genomic assembly, a previously owned version can be obtained following the directions below (to be run from within your `blast+_working_example` directory).

`cp /home/shared/biobootcamp/data/Lamellibrachia_luymesi_finished_genomic_transcriptomic_assemblies/Lamellibrachia_luymesi_all_genomic_RAY_05_2015.fasta .`

7. Leaving your file named `Contigs.fasta` is not very informative, bad practice and will lead to confusion when you happen upon it 6 months from now. So rename the file to something more descriptive using the mv command:

    i. `mv Contigs.fasta Lamellibrachia_luymesi_all_genomic_RAY_05_2018.fasta`

8. Generate some basic and advanced statistics for your assembly using the script `get_fasta_stats.pl`:

    i. `get_fasta_stats.pl -h` will provide you with options that can be supplied to the script in order to tailor the output as desired. It is suggested that the `-g` and `-T` options be used at a minimum. Also recall from the homework that the output from the script can be *redirected* to a file using `>`
    b. Draft your get_fasta_stats.pl command by hand on scratch paper with 1) included options and 2) a redirect to an appropriately named file.

9. Once you have drafted your `get_fasta_stats.pl` commands, execute it in your linux shell within the `blast+_working_example` directory where your `Lamellibrachia_luymesi_all_genomic_RAY_05_2018.fasta` file is located. Now do a long listing of the `blast+_working_example directory` contents to ensure that 1) the new output file exists and 2) is has a file size > 0 (otherwise they are empty). Remember that a "long listing" is `ls -al` from within the directory.

10. View the contents of your output file using `cat` or `less` and assess the assembly

Now that we have assessed some of the properties for our assemblies, let's identify whether large mitochondrial contigs or potentially a complete mt genome is present in the assembly. QUESTION: why hypothesize that large pieces of mt genome would be present in the assembly in the first place?

Part 2 --- Identify and recover large mitochondrial contigs from a genomic assembly

1. The sequences in your `Lamellibrachia_luymesi_all_genomic_RAY_05_2018.fasta` file will serve as subject (i.e., they will be the basis for the database that searches are conducted against). That means that we will need an appropriate query or queries to locate and "fish out" any potentially complete mitochondrial genome. While you can easily acquire such data from NCBI via a web browser, where is the fun in that? You can also do it from the command line in a number of ways. Here, we will be using a set of relatively new, powerful and unix-friendly utilities that are part of NCBI Edirect

(https://www.ncbi.nlm.nih.gov/books/NBK179288).

i. Type `esearch -help` and `efetch -help` to get an idea of the general usage – we can essentially perform an Entrez query from the command line. e.g. `esearch -db pubmed -query "Mcclintock B [AUTH]"`
ii. Try the command esearch `-db nuccore -query "Lamellibrachia luymesi [organism]"` Again the output is not particularly informative but note the 'count' field in the xml output. Does this value change if you leave out the query qualifier "organism"? What explains the difference?
iii. The command `efetch` is the key to retrieving and formatting your esearch results. Conveniently, `esearch` output can be "piped" into `efetch`: `esearch -db nuccore -query "Lamellibrachia luymesi [organism]" | efetch -format acc` Look back at the efetch help to see what this format means.
iv. That seems useful but what we want here is to repeat the command above with the format set to fasta. Redirect the output (hint: >) of the combined commands to a new file named something like `L_luymesi_genbank.fasta`

2. Your fasta output is relatively easy to scroll through, but imagine if it was 100x the size. Recall the required format of a fasta file and use grep to isolate the sequence headers:

i. Type `grep \> L_luymesi_genbank.fasta` (or whatever filename you chose). It looks like there are numerous potential mitochondrial sequences in our retrieval including multiple COI sequences.
ii. Repeat your grep search using "COI" as a search term note: you won't need the escape character '\', why not? Also try `less L_luymesi_sequences.fasta` and search within for COI using '/' and typing "COI".
iii. There are many elegant ways to pull out an individual COI sequence, let's make it easy for now and simply copy one of the COI seqeunces returned in the output of the `less` command and paste it into a separate, new, clearly named fasta file using `nano`.

3. We can now build our subject database from the `Lamellibrachia_luymesi_all_genomic_RAY_05_2018.fasta` file using `makeblastdb`.

i. Type the command `module load blast+/2.3.0` to load the BLAST+ suite into our current workspace in interactive mode.
ii. `makeblastdb -help` (provides you with options for constructing your BLAST+ DB) **NOTE**: you will see the there are both "required" and "optional" arguments to the makeblastdb program. As you might guess, the ones in the former category are mandatory and must be supplied while those in the latter are supplementary. Read through the list of "optional" arguments and consider using some in your command.
iii. Draft your `makeblastdb` command with 1) all "required" and 2) any "optional" arguments out ahead of time. Consult with members of your group on the arguments that you utilized and why you are using them.

4. Next, since we will be conducting a nucleotide query to nucleotide database search, `blastn` will be utilized.

    i. `blastn -help` (provides you with options for conducting your blastn search) **NOTE**: unlike `makeblastdb`, all arguments to `blastn` are "optional". Read through the list of "optional" arguments carefully and start to identify ones you might consider essential. Some bare minimum arguments you should consider are `-query`, `-out`, and `-db`.
    ii. Draft your `blastn` command with any and all "optional" arguments on paper or in a text editor on your own computer. Consult with members of the group on the arguments that you are utilized and again why you are using them. Make sure to name your output file with a `.blastout` file extension.

5. Once you have drafted your `makeblastdb` and `blastn` commands, add them to the appropriate section of the `BLAST+_example.sh` script using `nano`. Save the script and exit from `nano`.

6. Now submit your script to the ASC queue system using the directions at the bottom of the script.

7. Monitor this job using `squeue`.

Once we have a report (i.e., the `*.blastout` file) from blastn, we will want to parse it into a format that is MUCH more human readable.

Part 3 --- Parsing the output from BLAST+

1. Once `squeue` has cleared (i.e., indicating that the `makeblastdb` command and `blastn` search have successfully completed), check the contents of the `blast+_working_example directory` with a "long listing" (`ls -l`). You should see a few new files:

    i. `*.oXXXXXX` (the job summary; check this in case your job appears to have run into an error and died)
    ii. `*.blastout` (the blastn report itself containing the actual results of the search)
    iii. files ending in `*.nhr`, `*.nin` and `*.nsq` (these three (3) files represent the actual BLAST+ database itself. All three are required for our BLAST+ DB to be valid and functional. Keep this in mind if you construct BLAST+ DBs in one location and then move them to another).

2. Time to parse the `*.blastout` file into an easily accessible table using a script called `blast2table2.pl`:

    a. `blast2table2.pl -help` (provides the options for parsing your `blastn` report with

`blast2table2.pl`)
b. Draft the `blast2table2.pl` command with your selected options ahead of time. Discuss with members of your group the arguments that you are using and why you are using them. **HINT**: consider minimally `-format 9` , `-verbose` and a > redirect to send the results to an output file with a ".blastout.parsed" file extension.

3. Once you have drafted your `blast2table2.pl` command, execute it in the CLI within the `blast+_working_example` directory. Output from the parsing script should scroll by on the screen. A "long listing" (`ls -l`) in the `blast+_working_example` directory afterward should reveal a `*.blastout.parsed` file whose size is > 0.

4. Carefully examine the first 10 lines of the `*.blastout.parsed` file using the `head` command (remember this from the pre-Bootcamp assignment?). The header information for each of the columns can be found in the description of `-format 9` in `blast2table2.pl -help` .

5. Of the 15 fields (columns) of the `-format 9` output of `blast2table2.pl` , #11 seems promising if we want to:

    i. quantify how many contigs from our assembly have homology to the chosen queries (i.e., distinct mitochondrial COI gene copies from Lamellibrachia luymesi) and
    ii. the specific identities of those contigs in the NCBI+ DB (and by extension the original *.fasta file that we used to create the DB).

6. Let's extract field #11 of the report and identify how many unique entries occur (i.e., # of contigs from the assembly); be sure to provide **your** actual `*.blastout.parsed` name to the following command when executed in bash:

    i. `cat <filename>.blastout.parsed | awk -F "\t" '{print $11}' | sort | uniq`
    --- What just happened? Determine **as a group** exactly what the above pipeline accomplished. Remember that `man cmd_name` provides detailed information for any command as well as its options (another one from the pre-Bootcamp assignment). Don't worry, you'll get more exposure to commands and pipelines like the above in your very near future …

7. Repeat the above command, redirecting output to a file named `lure.txt` and examine the results with `cat` . How many unique entries (i.e., contig names) are in it? Is this the same across other members of the group? Let's use the contents of this file to fish out **entire** FASTA nucleotide entries from the `Lamellibrachia_luymesi_all_genomic_RAY_05_2018.fasta` file for each and every descriptor in the `lure.txt` file.

    i. `select_contigs.pl -h` (provides options for parsing a FASTA report with this particular script). For the exercise, consider 'lure.txt' as analogous to the 'select_file' mentioned in

the help info of `select_contigs.pl`. Therefore the `-n` flag should be specified.

ii. Draft the `select_contigs.pl` command with all selected options out in your own editor. Name your final output file with a `<filename>.potential_mito_genome.fasta` extension.

8. Once you have crafted your `select_contigs.pl` command with appropriate options, execute it on ASC. Check the final "*.potential_mito_genome.fasta" file the `get_fasta_stats.pl` script in #8 above for its statistics. Only a *single* contig of <~15K bp should be included in the final output file.

KEEP THIS FILE SINCE IT WILL BE USED LATER AS PART OF AN EXERCISE IN MITOCHONDRIAL GENOME ANNOTATION.

Tips

Tired of scrolling back through pages of help screens or program output? Pipe your command to the page-viewer `less`
e.g. `get_fasta_stats.pl -h | less` – as usual, type `q` to exit this type of pager.

https://github.com/NCBI-Hackathons/EDirectCookbook is a good resource for NCBI edirect utilities. Also see https://dataguide.nlm.nih.gov/eutilities/utilities.html#esearch and