Assessing Trinity assembly quality

Check out the Trinity assembly

You should have the assembly written to `trinity_out_dir.Trinity.fasta`. Check out the first few lines of the assembly: `$ head trinity_out_dir.Trinity.fasta`

You will see the first few transcripts. Note that g1 refers to gene 1 and i1 refers to isoform 1. There can be many isoforms per gene. This can become important in downstream applications such as orthology assessment or differential expression.

Now check how many transcripts were assembled. An easy way to do this is to count the number of `>` in the fasta file. These each correspond to a transcript. You can do this with `grep`.
`$ grep -c '>' trinity_out_dir.Trinity.fasta`

Contig stats

Now we will generate stats about the transcripts. you can do this with the `TrinityStats.pl` script. This will be a very short job.

Again open QSubGen

- Choose `short` for `CPU time`.
- Leave `Memory` at 1GB.
- Leave `serial` and `sh` selected.
- Begin typing `trinity` into the module field and select `bioinformatics/trinity/2.6.6`
- Fill in `Job specific commands` with: `TrinityStats.pl trinity_out_dir.Trinity.fasta`
- Choose a reasonable job name
- Copy and paste the script into your Unix text editor (hint: `nano` works well)
- Save the job as `trinity_stats.job`
- Submit the job with `qsub trinity_stats.job`
- When the job is complete, view the log file. You will see scores like contig N50, etc.

The N50 stats may not be the most useful for transcriptome analyses. They are dependent on both the length of the transcripts and can be influenced by the amount of expression of those particular genes. After we do transcript counting, we will generate ExN50 stats, which can be more informative. Aside from quick statistics like N50, we can generate more useful information about the quality of the transcriptome. We will try two of these methods next.

Generate "more useful" stats

> Following the command examples from the Trinity website here

Representation of reads

During this step, we will map some of the reads that we used in the assembler back to the transcriptome assembly. By evaluating how well paired end reads map back to the assembly, we can get a rough estimate of how well our assembly worked. To do this step, we will use the mapper `bowtie2` that is loaded with the Trinity module.

Let's open the familiar QSubGen. This time, you will fill it out yourself. Leave memory at the default and choose 10 CPU threads and load the `bioinformatics/trinity/2.6.6` module.

First, build a bowtie2 index for the transcriptome:

```
bowtie2-build trinity_out_dir.Trinity.fasta RNA_eye_assembly
```

Save the job file as `bowtie2_build.job`, then submit it.

This will create an index for your transcriptome that bowtie2 will you to run the read mapping in the subsequent step.

Second, to perform the alignment paired-end reads to capture the read alignment statistics we will run:

```
bowtie2 --threads $NSLOTS -q --no-unal -k 20 \
-x RNA_eye_assembly -1 data/RNA_Eye_1.fastq \
-2 data/RNA_Eye_2.fastq | samtools view -@10 -Sb -o bowtie2.bam
```

This command is using bowtie2 to run the alignment, which is in bam format. We then use samtools to assess the alignment.

Save the job file as `bowtie2.job`, then submit it.

> Hint: both of these commands can be combinded in a single job file.

Let's examine the statistics for our assembly, which are written in the last lines of our log file:

> Hint use `cat` or `tail` to read the last 15 or so lines of the log file.

```
  10000 (100.00%) were paired; of these:
    9093 (90.93%) aligned concordantly 0 times
    892 (8.92%) aligned concordantly exactly 1 time
    15 (0.15%) aligned concordantly >1 times
    ----
    9093 pairs aligned concordantly 0 times; of these:
      342 (3.76%) aligned discordantly 1 time
    ----
    8751 pairs aligned 0 times concordantly or discordantly; of these:
      17502 mates make up the pairs; of these:
        17379 (99.30%) aligned 0 times
        101 (0.58%) aligned exactly 1 time
        22 (0.13%) aligned >1 times
13.11% overall alignment rate
```

As you can see here, only 13.11% of the reads aligned properly to the assembled the transcriptome. 3.76% of the reads were 'aligned discordantly' (improper pairs). This means that each end of the paired end reads ended up on different contigs. This is an indication that the assembly is quite low quality and/or fragemented. In this case, this is because we used a subsample of the original data in this course. If we were to use the whole data set, these statistics would likely be much higher. A normal Trinity assembly will have > 70% of the reads properly mapped (proper pairs: yielding concordant alignments 1 or more times to the reconstructed transcriptome) to the assembly. If your number is below this threshold, it is possible that you should sequence at a greater depth of coverage.

Assess number of full-length coding transcripts

We can also evaluate transcriptome assemblies based on the number of fully assembled coding transcripts. One way to do this is to BLAST the transcripts against a database of protein sequences. We will use a reduced version of SWISSPROT, which is in the `data` directory.

Create a job file for this step. You should choose 8 CPUs and the default memory. You will want to load the blast module.

In the command field enter:

```
blastx -query trinity_out_dir.Trinity.fasta \
         -db data/mini_sprot.pep -out blastx.outfmt6 \
         -evalue 1e-20 -num_threads $NSLOTS \
         -max_target_seqs 1 -outfmt 6
```

Copy this to a job file called `trinity_blastx.job` and submit it.

Your job will create an output file called `blastx.outfmt6`. We will use the `analyze_blastPlus_topHit_coverage.pl` script to generate a table that contains information for the number of transcripts that contain full length protein sequence.

Since this will be very fast, we can just enter it into the command line:

```
$ module load bioinformatics/trinity
$ analyze_blastPlus_topHit_coverage.pl blastx.outfmt6 trinity_out_dir.Trinity.fasta d
ata/mini_sprot.pep | column -t
```

The output will look something like this:

```
#hit_pct_cov_bin  count_in_bin  >bin_below
100               2             2
90                0             2
80                1             3
70                1             4
60                3             7
50                3             10
40                0             10
30                3             13
20                5             18
10                1             19
```

This tells us that 2 transcripts had were between 90 and 100% length, 0 were between 80 and 90%, etc. The far right column is a cumulative number, e.g. only 4 transcripts contain >70% of the protein sequence length. Again, this is likely because we used a very small subsample of the original data in this course.