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/2.1.1`
- 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

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 `bowtie_PE_separate_then_join.pl` script that is included in Trinity.

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/2.1.1` module.

The command will be:
`bowtie_PE_separate_then_join.pl \ --target trinity_out_dir.Trinity.fasta \ --seqType fq \ --left data/wt_SRR1582651_1.fastq \ --right data/w`
Save the job file as `trinity_bowtie.job`, then submit it.

Since it uses the bowtie read aligner, it will write most of the output to the `bowtie_output` directory. You can look at the files generated with

```
$ ls -lh bowtie_out
```

One of the output files will be a .bam file with the reads mapped. We will evaluate how well the reads mapped to the assembly with the script `SAM_nameSorted_to_uniq_count_stats.pl`.

Create another job file using QSubGen. This time, we will load the Trinity module, and use the command:

```
SAM_nameSorted_to_uniq_count_stats.pl bowtie_out/bowtie_out.nameSorted.bam
```

Copy and paste the script and submit the job. Look at the log file. It should look something like this:

```
#read_type  count   pct
improper_pairs  3544    50.31
proper_pairs    2586    36.71
right_only  527 7.48
left_only   387 5.49

Total aligned reads: 7044
```

As you can see here, only 36.7% of the reads aligned properly to the assembled the transcriptome. 50% of the reads were 'improperly aligned'. This means that each end of the paired end reads ended up on different contigs. This is an indication that the assembly is quite fragemented. In this case, this is because we used a subsample of the original data or this workshop. 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 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 data/mini_sprot.pep | column -t
```

The output will look something like this:

```
#hit_pct_cov_bin  count_in_bin  >bin_below
100               79            79
90                17            96
80                11            107
70                18            125
60                16            141
50                22            163
40                36            199
30                42            241
20                64            305
10                24            329
```

This tells us that 79 transcripts had were between 90 and 100% length, 17 were between 80 and 90%, etc. The far right column is a cumulative number, e.g. 125 transcripts contain >70% of the protein sequence length.