Exercise 2: Post-assembly transcriptome analysis

In this exercise, we will analyze RNA-seq data from four samples from *Drosophila yakuba* (NCBI SRA SRP021207). They are from two different tissues (tis1 and tis2), with two biological replications for each tissue (rep1 and rep2). First, data from all 4 samples were combined and assembled by Trinity. In this exercise, you will not run the assembly, instead you will focus on post-assembly data analysis. You are provided with the assembly result file **`Trinity.fasta`**, together with 4 pairs of RNA-seq data files (one pair from each sample). The sample labels are: tis1rep1, tis1rep2, tis2rep1 and tis2rep2.

Part 1. Abundance Estimation using RSEM.

1. Create a working directory and copy all data files required for this workshop into the working directory. (Replace "**`MyUserID`**" with you login ID)

```
mkdir /workdir/MyUserID
cd /workdir/MyUserID
cp /shared_data/Trinity_workshop_2018/part2/* ./
export TRINITY_HOME=/programs/trinityrnaseq-Trinity-v2.8.4
```

Note: the last command (**export**) is to set up a Linux environment variable **TRINITY**. After it is set, you can use **$TRINITY** to replace the string **/programs/trinityrnaseq-2.2.0**. Every time you open a new session and want to use **$TRINITY**, you need to execute this command "**export TRINITY=…**", unless you include this line in the **.bash_profile** file in your home directory.

2. Create the following shell script. You can do it on your Windows Laptop using **Notepad++**, on a Mac – using **TextWrangler**. You can also create the file directly on your workshop Linux workstation, for example using the **nano** text editor (you can put the file in your home directory **`/home/MyUserID`**). Name the file **`quantify.sh`**. Make sure that each command is typed on a single line, or brake lines with the "\" character at the end of each part. The explanation of this shell script is in the note below. This step could take several hours, run it in "screen" session.

```
$TRINITY_HOME/util/align_and_estimate_abundance.pl --transcripts Trinity.fasta --est_method RSEM \
--aln_method bowtie2 --prep_reference

$TRINITY_HOME/util/align_and_estimate_abundance.pl --transcripts Trinity.fasta --seqType fq --aln_method
bowtie2 --est_method RSEM --SS_lib_type RF --thread_count 8 --trinity_mode --samples_file mysamples
```

Note:

a) The first command in this script will index the transcriptome sequence file **`Trinity.fasta`**, which is the assembled transcriptome and serves as reference for the transcript quantification. After indexing is done, fastq files from each sample can be aligned to the reference transcriptome.

b) The second command would run **`bowtie2`** to align reads from each sample to the reference, and run RSEM to quantify read counts for each gene/isoform. Intermediate and final results from these runs will be located in directory **`/workdir/MyUserID/quant_dir`** (as specified on the command lines). The sequencing data file names are specified in the file mysamples. The file format of the sample file is defined in the web page: https://github.com/trinityrnaseq/trinityrnaseq/wiki/Trinity-Transcript-Quantification

3. After it is done you would find one new directories for each sample: tissue1_rep1, tissue1_rep2, tissue2_rep1, tissue1_rep2. Within each directory, you would find a file RSEM.genes.results. The "expected_count" column is the "transcript count" for reach gene. You would use this column for further analysis. Use the following command to combine all samples into one data table. Please note I uses "cross_sample_norm none", as software like "EdgeR" and "DESeq2" expect un-normalized raw counts. The combined file name is: mystudy.isoform.counts.matrix. As your input is the gene count file, the file actually gives you gene level count.

```
$TRINITY_HOME/util/abundance_estimates_to_matrix.pl --est_method RSEM \

  --gene_trans_map none \

  --cross_sample_norm none \

  --out_prefix mystudy \

  --name_sample_by_basedir \

   tissue1_rep1/RSEM.genes.results \

   tissue1_rep2/RSEM.genes.results \

   tissue2_rep1/RSEM.genes.results \

   tissue2_rep2/RSEM.genes.results
```

Part 2. Evaluate assembled transcript with BUSCO

Instructions of running BUSCO is also available on BioHPC software page:
https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=255#c

```
cd /workdir/xxxxx   ## replace xxxxx with your user ID

cp /programs/busco-3.0.2/config/config.ini  ./

cp -r /programs/augustus-3.3/config ./

tar xvfz insecta_odb9.tar.gz

export AUGUSTUS_CONFIG_PATH=/workdir/xxxxx/config

export BUSCO_CONFIG_FILE=/workdir/xxxxx/config.ini

export PYTHONPATH=/programs/busco-3.0.2/lib/python3.6/site-packages

export PATH=/programs/busco-3.0.2/scripts:/programs/augustus-3.3/bin:/programs/augustus-3.3/scripts:$PATH

run_BUSCO.py --in ./Trinity.fasta --lineage_path ./insecta_odb9 --mode genome --out trinityBUSCO --cpu 4
```

After it is done, the result file is:
/local/workdir/qisun/run_trinityBUSCO/ short_summary_trinityBUSCO.txt

Note: the BUSCO lineage-specific database can be downloaded from BUSCO web site: https://busco.ezlab.org/ . When you work with your real data, you need to find a lineage that is closest to the species you are analyzing, In this case, you use insects.

Part 3. Evaluate assembled transcript by comparing with known proteins

The Trinity package provides a tool **`analyze_blastPlus_topHit_coverage.pl`** to evaluate the assembled transcripts by comparing them with known proteins. In this example, we will compare the assembly with the annotated *Drosophila melanogaster* proteins. A fasta file of all *melanogaster* proteins (**`Drosophila_melanogaster.BDGP5.pep.all.fa`**) is included among the exercise data files. If there is no closely related species, you can also use the **Uniprot** sequences for evaluation.
(ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.fasta.gz)

Create and run the following shell script in **`/workdir/MyUserID`**:

```
makeblastdb -in Drosophila_melanogaster.BDGP5.pep.all.fa -dbtype prot

blastx -query Trinity.fasta \
-db Drosophila_melanogaster.BDGP5.pep.all.fa \
-out blastx.outfmt6 -evalue 1e-20 -num_threads 4 \
-max_target_seqs 1 -outfmt 6

$TRINITY/util/analyze_blastPlus_topHit_coverage.pl \
blastx.outfmt6 Trinity.fasta Drosophila_melanogaster.BDGP5.pep.all.fa
```

The three commands executed by the script are:

1. **`makeblastdb`**: create a blast database from the D. melanogaster protein sequences;
2. **`blastx`**: run blastx against the D.melanogaster protein database;
3. `analyze_blastPlus_topHit_coverage.pl`: summarize the blast results, and check the how many full length proteins are covered in the assembly.

The output is the file **`blastx.outfmt6.hist`**. The interpretation of this file can be found at https://github.com/trinityrnaseq/trinityrnaseq/wiki/Counting-Full-Length-Trinity-Transcripts.

The Trinity web site (http://trinityrnaseq.github.io/#Downstream_analyses) provides detailed documentations for the tools we use in this workshop.