De novo transcriptome assembly using Trinity. 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_2015/part2/* ./

export TRINITY=/programs/trinityrnaseq-2.0.4
```

Note: the last command (**export**) is to set up a Linux environment variable **TRINITY**. After it is set, you can use **$TRINITY** to replace the string **/programs/trinityrnaseq-2.0.4**. Every time you open a new session and want to use **$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.

```
$TRINITY/util/align_and_estimate_abundance.pl --transcripts Trinity.fasta --est_method RSEM \
--aln_method bowtie --trinity_mode --prep_reference

$TRINITY/util/align_and_estimate_abundance.pl --transcripts Trinity.fasta --seqType fq --aln_method bowtie
--est_method RSEM --left tis1rep1-10M_ref_1.fastq.gz --right tis1rep1-10M_ref_2.fastq.gz --SS_lib_type RF
--thread_count 4 --trinity_mode --output_prefix tis1rep1

$TRINITY/util/align_and_estimate_abundance.pl --transcripts Trinity.fasta --seqType fq --aln_method bowtie
--est_method RSEM --left tis1rep2-10M_ref_1.fastq.gz --right tis1rep2-10M_ref_2.fastq.gz --SS_lib_type RF
--thread_count 4 --trinity_mode --output_prefix tis1rep2

$TRINITY/util/align_and_estimate_abundance.pl --transcripts Trinity.fasta --seqType fq --aln_method bowtie
--est_method RSEM --left tis2rep1-10M_ref_1.fastq.gz --right tis2rep1-10M_ref_2.fastq.gz --SS_lib_type RF
--thread_count 4 --trinity_mode --output_prefix tis2rep1

$TRINITY/util/align_and_estimate_abundance.pl --transcripts Trinity.fasta --seqType fq --aln_method bowtie
--est_method RSEM --left tis2rep2-10M_ref_1.fastq.gz --right tis2rep2-10M_ref_2.fastq.gz --SS_lib_type RF
--thread_count 4 --trinity_mode --output_prefix tis2rep2
```

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) Each of the following commands would run **`bowtie`** to align reads from each sample to the reference, and run RSEM to quantify read counts for each gene/isoform.

3. If you created the script on your laptop, use **FileZilla** to upload it to your home directory **`/home/MyUserID`**. If the file has been created on a Windows computer, make sure you run **`dos2unix quantify.sh`** to convert it to a Linux format. Then launch the script as follows:

```
cd /workdir/MyUserID

nohup sh /home/MyUserID/quantify.sh >& logfile &
```

The program will be running in the background, with all screen output saved to the file **`logfile`** (in **`/home/MuUserID`**). This step could take a few hours to finish.

4. After it is done the following files will be created. There are two files for each sample. The **`*gene.results`** files are read count per gene. The ***`isoforms.results`** are read count per isoforms:

`tis1rep1.genes.results`	**`tis1rep1. isoforms.results`**
`tis1rep2.genes.results`	**`tis1rep2. isoforms.results`**
`tis2rep1.genes.results`	**`tis2rep1. isoforms.results`**
`tis2rep2.genes.results`	**`tis2rep2. isoforms.results`**

Part 2. Identify differentially expressed genes between the two tissues.

1. Combine read count from all four samples into a matrix, and normalize the read count using the TMM method. This command will take in RSEM output files from each sample, and combine them into a single matrix file. You can run the following command directly, or, as done previously, prepare and run a shell script containing this command.

```
$TRINITY/util/abundance_estimates_to_matrix.pl \
--est_method RSEM  tis1rep1.genes.results tis1rep2.genes.results \
tis2rep1.genes.results tis2rep2.genes.results --out_prefix mystudy
```

   Note: This step can be performed at either the gene level or isoform level. For this exercise, we do all the analysis at gene level, so we will use the RSEM output files **`*gene.results`**. After this step, two matrix files will be created: **`mystudy.counts.matrix`** and **`mystudy.TMM.fpkm.matrix`**. In both files, each column represents a sample, and each row represents a gene, the values are either the raw read counts or normalized FPKM values. The "counts" file will be used for differentially expressed gene identification, and the "fpkm" file will be used for clustering analysis. By default, the fpkm file is normalized with TMM method.

2. Identify differentially expressed genes. We will need a text file describing the samples. This file should have two columns separated with <Tab>. The first column corresponds to your designated biological conditions, in this case, the two different tissues (tis1 and tis2). The second column contains the sample names, corresponding to those listed in the header line in the matrix file you created in the last step.

```
tis1    tis1rep1
tis1    tis1rep2
tis2    tis2rep1
tis2    tis2rep2
```

   You can create this file either by Excel (convenient for larger datasets; must be saved as a Tab delimited text file) or simply a text editor. For the purpose of the exercise, the file has already been created and is included among the exercise data files under the name **`mysamples`**.

   To identify differentially expressed genes with **edgeR**, run the following command:

```
$TRINITY/Analysis/DifferentialExpression/run_DE_analysis.pl \
--matrix mystudy.counts.matrix \
--method edgeR \
--samples_file mysamples \
--min_rowSum_counts 10 \
--output edgeR_results
```

   As in previous steps, you may find it convenient to create a shell script containing this command and run this script.

Note:

The tool **`run_DE_analysis.pl`** is a PERL script that calls **Bioconductor** package **edgeR**. One parameter required by this tool is **`min_rowSum_counts`**. In this example we set it to 10, which would remove genes with read counts less than 10 (read counts summed from all 4 samples). The output files are in the directory **`edgeR_results`**. Using a text editor, open he file **`mystudy.counts.matrix.tis1_vs_tis2.edgeR.DE_results`** (you can also download it to your laptop and open it in Excel). It provides several values for each gene. 1) FDR to indicate whether a gene is differentially expressed or not; 2) logFC is the log2 transformed fold change between the two tissues; 3) logCPM is the log2 transformed normalized read count of average of the samples. We normally filter this list to FDR <0.05 or below. To be more conservative, you could also use more stringent FDR cutoff (e.g. <0.001), and only keep genes with high logFC (e.g. <-2 and >2) and/or high logCPM (e.g. >1). In the **`edgeR_results`** directory there is also a "volcano plot" to visualize the distribution of the DE genes.

Part 3. Clustering analysis

The following script will do hierarchical clustering and k-means clustering for samples and genes. This analysis needs to be performed in the **`edgeR_results`** directory generated in Part 2. The clustering will be performed only on differentially expressed genes, with FDR and logFC cutoff defined by –P and -C parameters (as before, you may prefer to include the two **`$TRINITY`** commands in a script):

```
cd edgeR_results

$TRINITY/Analysis/DifferentialExpression/analyze_diff_expr.pl \
--matrix ../mystudy.TMM.fpkm.matrix --samples ../mysamples  -P 1e-3 -C 2 \
--output cluster_results

$TRINITY/Analysis/DifferentialExpression/define_clusters_by_cutting_tree.pl \
-K 6 -R cluster_results.matrix.RData
```

In this example, we set K=6 for k-means analysis. The genes will be separated into 6 groups based on expression pattern, with expression patterns of each group shown in a PDF file. There are two pre-filtered files produced: **`*DE_results.P1e-3_C2.tis1-UP.subset`** and **`*DE_results.P1e-3_C2.tis2-UP.subset`**, with differentially expressed genes (FDR cutoff 0.001, logFC cutoff 2 and -2).

Part 4. Function annotation

As the function annotation step is computationally expensive (100k transcripts could take about two to three days on a BioHPC lab general computer), we are not going to run it for this exercise. Instead, you are provided with output files from function annotation pipeline: **`go_annotations.txt`** and **`trinotate_annotation_report.xls`**.

As you will need to perform this step when doing your own projects, here is the instruction of running function annotation.

Trinity authors recommend to use **Trinotate** (http://trinotate.github.io/) to do function annotation on the assembled transcripts. The step-by-step guidance of running **Trinotate** on BioHPC lab computers is provided at our web site: https://cbsu.tc.cornell.edu/lab/userguide.aspx?a=software&i=143#c .

If your **`Trinity.fasta`** file has too many sequences, **Trinotate** could take very long time to finish. One option is to filter the **`Trinity.fasta`** file before running **Trinotate**. For example, you could remove all isoforms that are extremely low expressed. Trinity provides a tool **`filter_fasta_by_rsem_values.pl`** to do this. You can filter by **TPM**, **FPKM**, or **IsoPct** (where '**TPM**' stands for **T**ranscripts **P**er **M**illion, '**FPKM**' stands for **F**ragments **P**er **K**ilobase of transcript per **M**illion mapped reads, and '**IsoPct**' stands for **Iso**form **P**er**C**en**T**age - the percentage of this transcript's abundance over its parent gene's abundance). In this example, all isoforms that have FPKM<5, TPM<10, or IsoPct<5% are removed (any one criteria matched in any one of the samples):

```
$TRINITY/util/filter_fasta_by_rsem_values.pl \
--rsem_output=tis1rep1.isoforms.results,tis2rep1.isoforms.results,\
tis1rep2.isoforms.results,tis2rep2.isoforms.results \
--fasta=Trinity.fasta --output=Trinity.filtered.fasta --isopct_cutoff=5 \
--fpkm_cutoff=10 --tpm_cutoff=10
```

Alternatively, **BLAST2GO** is another commonly used tool for function annotation and enrichment test. Here is the instruction of running **BLAST2GO** for function annotation: http://cbsu.tc.cornell.edu/lab/doc/instruction_blast2go.htm . **BLAST2GO** provides its own enrichment test tool. You can read the documentation at the **BLAST2GO** web site, or use our RNA-seq workshop handout: http://cbsu.tc.cornell.edu/lab/doc/RNA-Seq-2015-02-exercise3.pdf

Part 5. Gene Ontology Enrichment analysis

Trinity provides a tool (script **`run_GOseq.pl`**) to do enrichment analysis. This tool uses the **Bioconductor** library **goseq** for the actual analysis. The tool requires the GO annotation file, list of DE genes for testing. It also requires length of all genes to correct RNA-seq bias due to gene length. The output is a spreadsheet named **`*GOseq.enriched`**, which list the GO categories that are enriched in your DE gene set.

You can also perform this test on gene list generated from K-means clustering analysis.

To do this analysis, we need to prepare the input files. For the DE gene list you can simply use the pre-filtered DE gene list (file **`mystudy.counts.matrix.tis1_vs_tis2.edgeR.DE_results.P1e-3_C2.tis1-UP.subset`**) in the **`edgeR_results`** directory. The GO annotation file is from the **Trinotate**, the **`go_annotations.txt`** file provided with the exercise data files. We can generate the gene length using one of the RSEM output **`*.genes.results`** files.

```
cd /workdir/MyUserID

mkdir enrichment

cd enrichment

ln -s ../edgeR_results/mystudy.counts.matrix.tis1_vs_tis2.edgeR.DE_results.P1e-3_C2.tis1-UP.subset .

ln -s ../tis1rep1.genes.results .

ln -s ../go_annotations.txt .

cat tis1rep1.genes.results | cut -f 1,3 > genes.lengths.txt

$TRINITY/Analysis/DifferentialExpression/run_GOseq.pl \
--genes_single_factor mystudy.counts.matrix.tis1_vs_tis2.edgeR.DE_results.P1e-3_C2.tis1-UP.subset \
--GO_assignments  go_annotations.txt \
--lengths genes.lengths.txt
```

Note:
The **`ln -s`** commands create **symbolic links** to all the files that we need for this step. Symbolic links (equivalents of "shortcuts" on Windows) provide a way to traet a file as local to the current directory without having to copy or move the file over from its original location.
The **`cat`**…**`| cut`** command takes columns 1 and 3 from file **`tis2rep1.genes.results`** and writes them into a new file named **`genes.lengths.txt`**.

The last command, **`run_GOseq.pl`**, performs the enrichment analysis.

Part 6. 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:

```
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 http://trinityrnaseq.github.io/analysis/full_length_transcript_analysis.html .

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