A Tutorial: De novo RNA-Seq Assembly and Analysis Using Trinity and EdgeR

The following details the steps involved in:

- Generating a Trinity *de novo* RNA-Seq assembly
- Mapping reads and Trinity transcripts to a reference genome
- Visualizing the aligned reads and transcripts in comparison to reference transcript annotations.
- Identifying differentially expressed transcripts using EdgeR and various Trinity-included helper utilities.

All required software and data are provided pre-installed on a VirtualBox image. See companion 'Rnaseq_Workshop_VM_installation.pdf' for details. Data content and environment configurations are described therein and referenced below.

Before Running:

After installing the VM, be sure to quickly update the contents of the rnaseq_workshop_data directory by:

```
% cd rnaseq_workshop_data

% svn up
```

This way, you'll have the latest content, including any recent bugfixes.

Automated and Interactive Execution of Activities

To avoid having to cut/paste the numerous commands shown below into a unix terminal, the VM includes a script 'runTrinityDemo.pl' that enables you to run each of the steps interactively. To begin, simply run:

```
% ./runTrinityDemo.pl -I --DE
```

The -I parameter indicates to run interactively, and --DE indicates to include the differential expression analysis activities.

De novo assembly of reads using Trinity

To generate a reference assembly that we can later use for analyzing differential expression, first combine the read data sets for the different conditions together

into a single target for Trinity assembly. Combine the left reads and the right reads of the paired ends separately like so:

% cat condA.left.fa condB.left.fa > both.left.fa
% cat condA.right.fa condB.right.fa > both.right.fa

Now run Trinity:
(reminder, we refer to $TRINTY_HOME/ as the installation directory for the Trinity software and included utilities).

% $TRINITY_HOME/ Trinity.pl --seqType fa --left both.left.fa --right both.right.fa --CPU 4 --JM 4G

The assembled transcripts will be found at 'trinity_out_dir/Trinity.fasta'.

Just to look at the top few lines of the assembled transcript fasta file, you can run:

% head trinity_out_dir/Trinity.fasta

Examine assembly stats

Capture some basic statistics about the Trinity assembly:

% $TRINITY_HOME/util/TrinityStats.pl trinity_out_dir/Trinity.fasta

Total trinity transcripts: 621
Total trinity components: 620
Contig N50: 1204

Compare de novo reconstructed transcripts to reference annotations

Since we happen to have a reference genome and a set of reference transcript annotations that correspond to this data set, we can align the Trinity contigs to the genome and examine them in the genomic context.

a. Align the transcripts to the genome using GMAP

First, prepare the genomic region for alignment by GMAP like so:

% gmap_build -d genome -D . -k 13 genome.fa

Now, align the Trinity transcript contigs to the genome, outputting in SAM format, which will simplify viewing of the data in our genome browser.

% gmap -D . -d genome trinity_out_dir/Trinity.fasta -f samse > trinity_gmap.sam

Convert to a coordinate-sorted BAM (binary sam) format like so:

% samtools view –Sb trinity_gmap.sam > trinity_gmap.bam
% samtools sort trinity_gmap.bam trinity_gmap

Now index the bam file to enable rapid navigation in the genome browser:

% samtools index trinity_gmap.bam

b. Align RNA-seq reads to the genome using Tophat

Next, align the combined read set against the genome so that we'll be able to see how the input data matches up with the Trinity-assembled contigs. Do this by running TopHat like so:

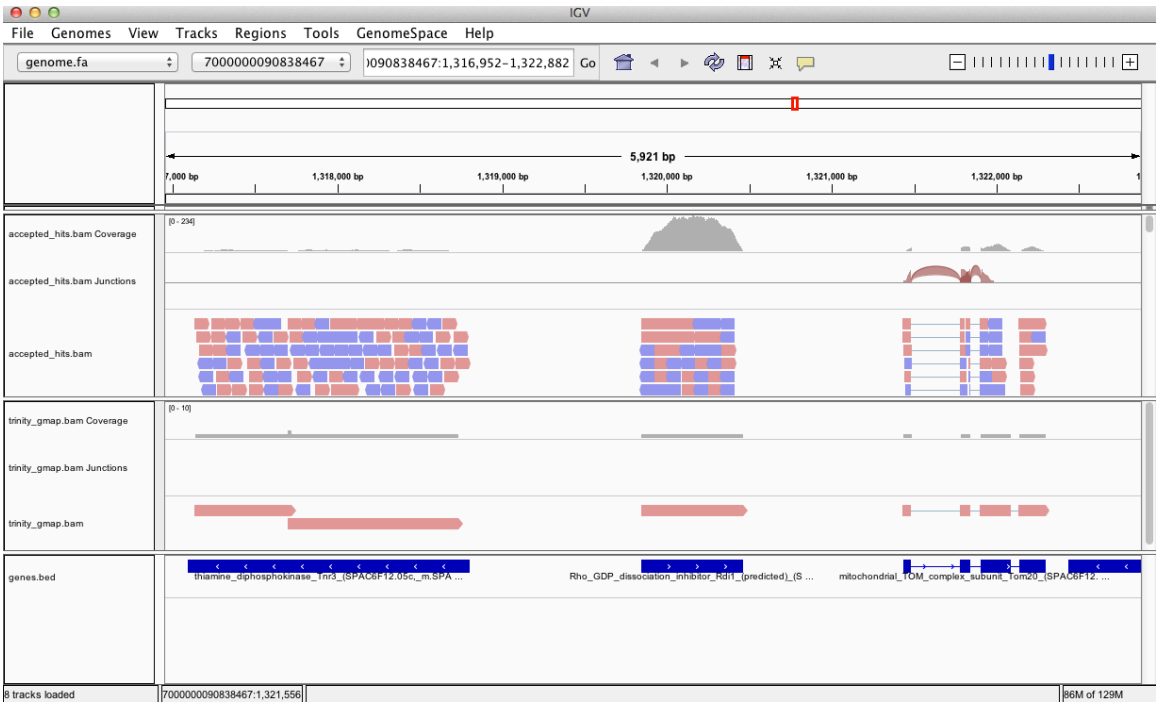
prep the genome for running tophat
% bowtie-build genome.fa genome

now run tophat:
% tophat -I 1000 -i 20 genome both.left.fa both.right.fa

index the tophat bam file needed by the viewer:
% samtools index tophat_out/accepted_hits.bam

c. Visualize all the data together using IGV

% java -jar $IGV/igv.jar -g `pwd`/genome.fa
`pwd`/genes.bed,`pwd`/tophat_out/accepted_hits.bam,`pwd`/trinity_gmap.bam



Does Trinity fully or partially reconstruct transcripts corresponding to the reference transcripts and yielding correct structures as aligned to the genome?

Are there examples where the de novo assembly resolves introns that were not similarly resolved by the alignments of the short reads, and vice-versa?

Abundance estimation using RSEM

To estimate the expression levels of the Trinity-reconstructed transcripts, we use the strategy supported by the RSEM software. We first align the original rna-seq reads back against the Trinity transcripts, then run RSEM to estimate the number of rna-seq fragments that map to each contig. Because the abundance of individual transcripts may significantly differ between samples, the reads from each sample must be examined separately, obtaining sample-specific abundance values.

For the alignments, we use 'bowtie' instead of 'tophat'. There are two reasons for this. First, because we're mapping reads to reconstructed cDNAs instead of genomic sequences, properly aligned reads do not need to be gapped across introns. Second, the RSEM software is currently only compatible with gap-free alignments.

The RSEM software is wrapped by scripts included in Trinity to facilitate usage in the Trinity framework.

a. Quantification for condition-A reads.

The following script will run RSEM, which first aligns the RNA-Seq reads to the Trinity transcripts using the Bowtie aligner, and then performs abundance estimation.

```
% $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl --seqType fa \
      --left condA.left.fa --right condA.right.fa \
      --transcripts trinity_out_dir/Trinity.fasta \
      --prefix condA -- --no-bam-output
```

Once finished, RSEM will have generated two files: 'condA.isoforms.results' and 'condA.genes.results'. These files contain the Trinity transcript and component (the Trinity analogs to Isoform and gene) rna-seq fragment counts and normalized expression values.

b. Quantification for condition-B reads

Now, perform the same operation for condition-B:

```
% $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl --seqType fa \
      --left condB.left.fa --right condB.right.fa \
      --transcripts trinity_out_dir/Trinity.fasta \
      --prefix condB -- --no-bam-output
```

Note the new output files generated: 'condB.isoforms.results' and 'condB.genes.results'.

Differential Expression Using EdgeR

To run edgeR and identify differentially expressed transcripts, we need a data table containing the raw rna-seq fragment counts for each transcript and sample analyzed. We can combine the RSEM-computed isoform fragment counts into a matrix file like so:

```
# merge them into a matrix like so:
% $TRINITY_HOME/ util/RSEM_util/merge_RSEM_frag_counts_single_table.pl
condA.isoforms.results condB.isoforms.results > transcripts.counts.matrix

# before running edgeR, we need the transcript length information, which we can
extract from one of the RSEM.isoforms.results files like so:
% cat condA.isoforms.results | cut -f1,3,4 > trans_lengths.txt

# now, run edgeR via the helper script provided in the Trinity distribution:
% $TRINITY_HOME/ Analysis/DifferentialExpression/run_DE_analysis.pl --matrix
transcripts.counts.matrix --method edgeR --no_eff_length --output edgeR
```

Examine the contents of the edgeR/ directory.

% ls edgeR/

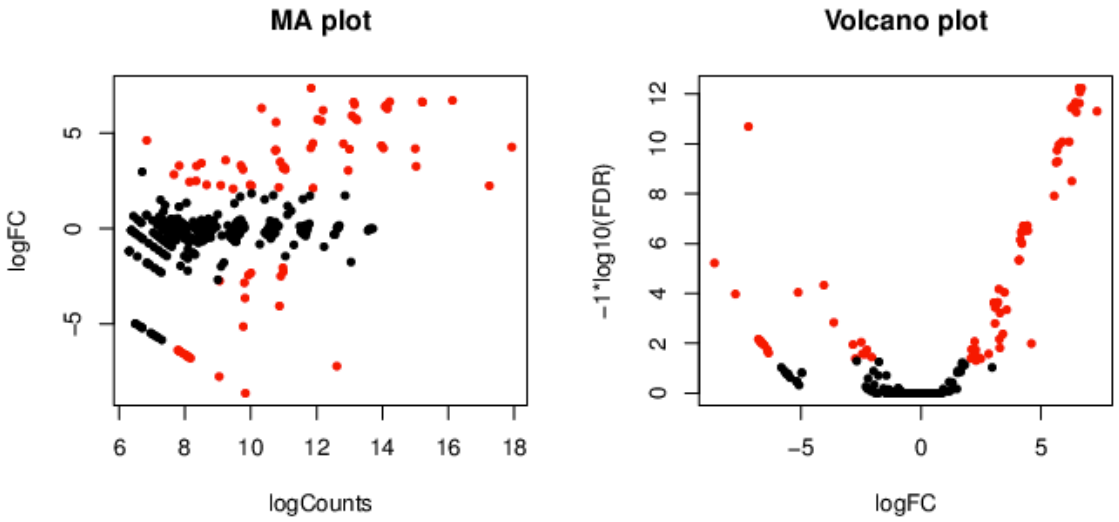
The file 'transcripts.counts.matrix.condA_vs_condB.edgeR.DE_results' contains the output from running EdgeR to identify differentially expressed transcripts. Examine the format of this file:

% head edgeR/transcripts.counts.matrix.condA_vs_condB.edgeR.DE_results

logFC	logCPM	PValue	FDR
comp5593_c0_seq1	6.69800714119315	16.1249761736403	1.75816549640362e-15
comp844_c0_seq1	6.63057977003254	15.1976845446146	4.06529730820251e-15
comp1335_c0_seq1	6.60888607985679	15.2161403709572	4.56157354999284e-15
comp20247_c0_seq1	6.62991619131932	14.2195834388687	8.95858961048408e-15

These data include the log fold change (logFC), log counts per million (logCPM), P value from an exact test, and false discovery rate (FDR).

The EdgeR analysis above generated both MA and Volcano plots based on these data. See file 'transcripts.counts.matrix.condA_vs_condB.edgeR.DE_results.MA_n_Volcano.pdf' as shown below:



How many differentially expressed transcripts do we identify if we require the FDR to be at most 0.05? You could import the tab-delimited text file into your favorite spreadsheet program for analysis and answer questions such as this, or we could run some unix utilities and filters to query these data. For example, a unix'y way to answer this question might be:

```
%  sed '1,1d' transcripts.counts.matrix.condA_vs_condB.edgeR.DE_results | awk '{ if ($5 <= 0.05) print;}' | wc -l
    74
```

Trinity includes a script to further facilitate analysis of these data, extracting transcripts that are above some statistical significance (FDR threshold) and fold-change in expression, and generating figures such as heatmaps and other useful plots, as described below.

TMM normalization followed by expression profiling

Before we begin to examine patterns of expression across multiple samples, we need to first normalize the FPKM expression values across samples, which will account for differences in RNA composition (ex. highly expressed transcripts in one or more samples that skew the relative proportions of transcripts in each sample). Here, we apply TMM normalization (see: http://genomebiology.com/2010/11/3/r25) to generate a matrix of normalized FPKM values across all samples, like so:

```
% $TRINITY_HOME/Analysis/DifferentialExpression/run_TMM_normalization_write_FPKM_matrix.pl --matrix transcripts.counts.matrix --lengths trans_lengths.txt
```

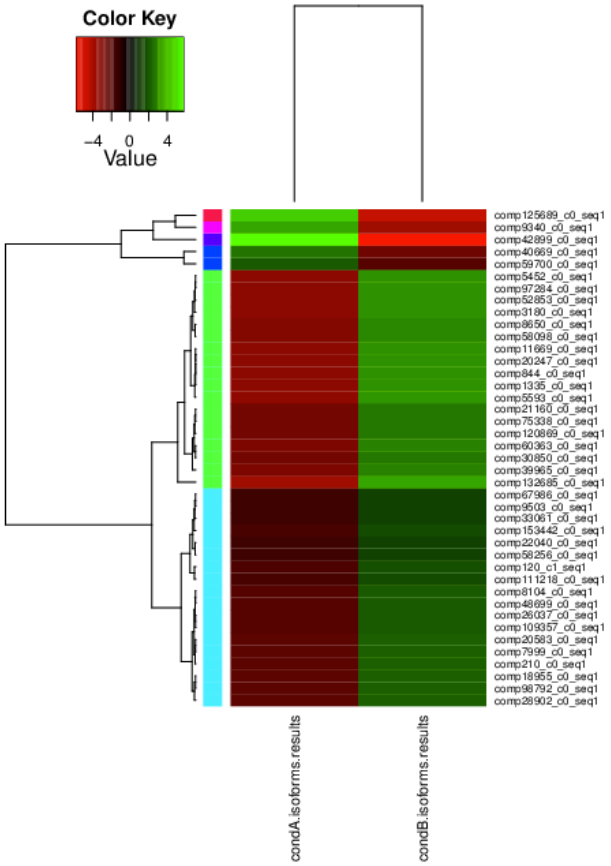
The file 'transcripts.counts.matrix.TMM_info.txt' includes the results from running the TMM normalization step, and the new 'effective' library sizes (depth of read sequencing) are indicated. These adjusted library sizes are used to recompute the FPKM expression values, as provided in the file 'transcripts.counts.matrix.TMM_normalized.FPKM'. Although the raw fragment counts are used for differential expression analysis, the normalized FPKM values are used below in examining profiles of expression across different samples, and are shown in heatmaps and related expression plots.

Extracting differentially expressed transcripts and generating heatmaps

Extract those differentially expressed transcripts that are at least 4-fold differentially expressed at a significance of <= 0.001

```
% $TRINITY_HOME/Analysis/DifferentialExpression/analyze_diff_expr.pl --matrix matrix.TMM_normalized.FPKM
```

This will generate a clustered heatmap file called 'diffExpr.P0.001_C2.matrix.heatmap.pdf' as shown below, with transcripts clustered along the vertical axis and samples clustered along the horizontal axis.



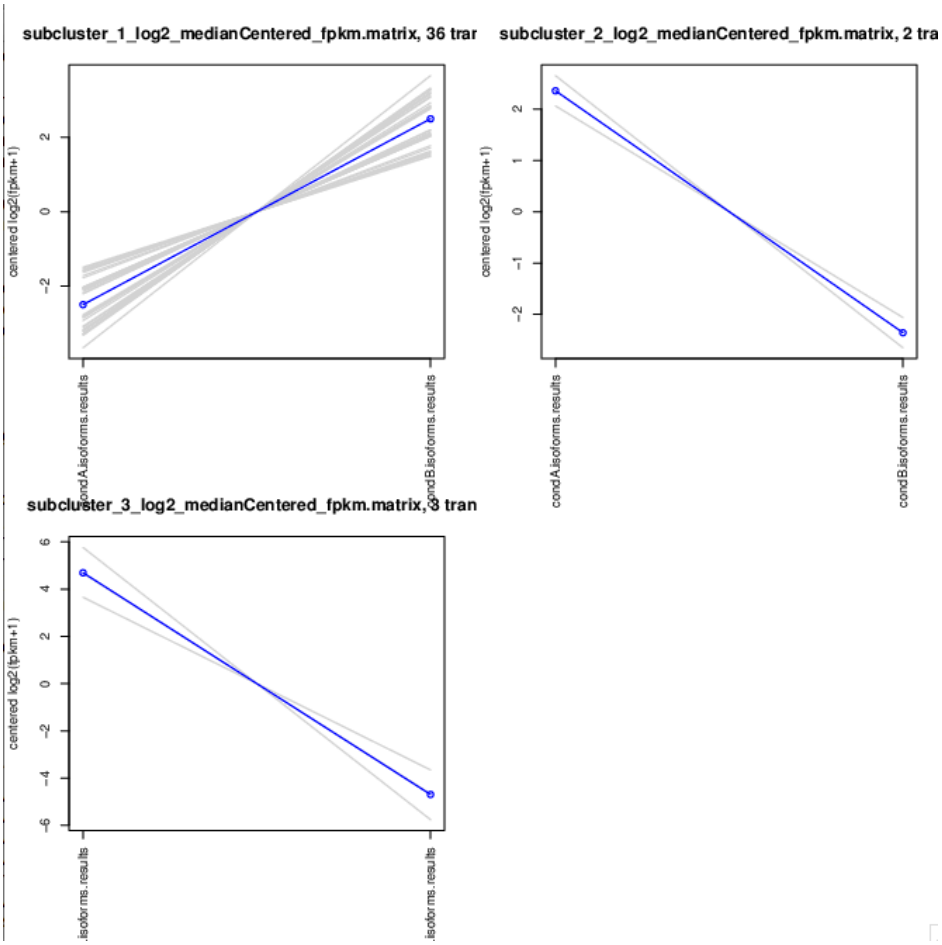
Extract transcript clusters by expression profile by cutting the dendrogram

Extract clusters of transcripts with similar expression profiles by cutting the transcript cluster dendrogram at a given percent of its height (ex. 25%), like so:

```
%
$TRINITY_HOME/Analysis/DifferentialExpression/define_clusters_by_cutting_tree.pl --Ptree 25 -R diffExpr.P0.001_C2.matrix.R.all.RData
```

This creates a directory containing the individual transcript clusters, including a pdf file that summarizes expression values for each cluster according to individual charts:

See:
diffExpr.P0.001_C2.matrix.R.all.RData.clusters_fixed_P_25/my_cluster_plots.pdf



More information on Trinity and supported downstream applications can be found from the Trinity software website: http://trinityrnaseq.sf.net