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

(updated: 2014-10-21)

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_2014

% svn up
```

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

Data Content:

This demo uses RNA-Seq data corresponding to Schizosaccharomyces pombe (fission yeast), involving paired-end 76 base strand-specific RNA-Seq reads corresponding to four samples: Sp_log (logarithmic growth), Sp_plat (plateau phase), Sp_hs (heat shock), and Sp_ds (diauxic shift).

There are 'left.fq' and 'right.fq' FASTQ formatted Illlumina read files for each of the four samples. Also included is a 'genome.fa' file corresponding to a genome sequence, and annotations for reference genes ('genes.bed' or 'genes.gff3').

Note, although the genes, annotations, and reads represent genuine sequence data, they were artificially selected and organized for use in this tutorial, so as to provide varied levels of expression in a very small data set, which could be processed and analyzed within an approximately one hour time session and with minimal computing resources.

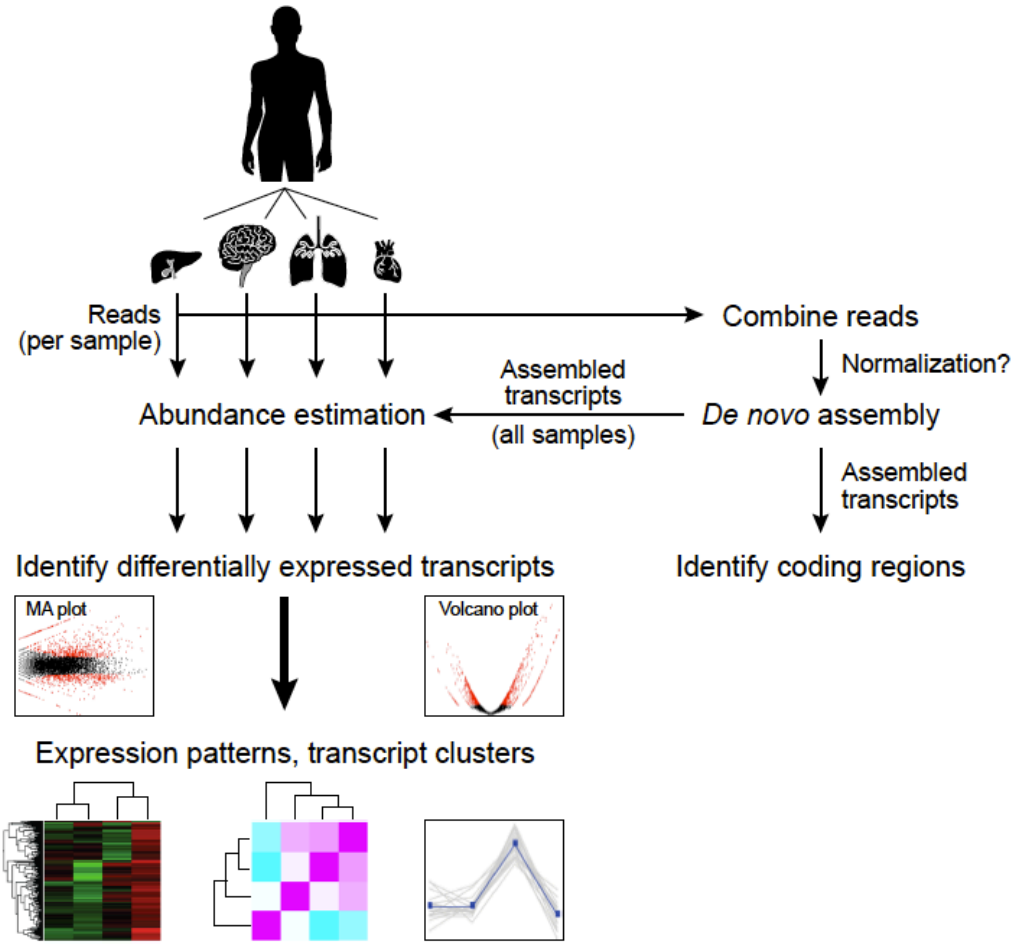
Automated and Interactive Execution of Activities

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
```

Note, by default and for convenience, the demo will show you the commands that are to be executed. This way, you don't need to type them in yourself.

The protocol followed is that described here:
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3875132/



Below, we refer to $TRINITY_HOME/ as the directory where the Trinity software is installed.

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 Sp_ds.left.fq Sp_hs.left.fq Sp_log.left.fq Sp_plat.left.fq > ALL.LEFT.fq

% cat Sp_ds.right.fq Sp_hs.right.fq Sp_log.right.fq Sp_plat.right.fq > ALL.RIGHT.fq

Now run Trinity:

% $TRINITY_HOME/Trinity --seqType fq --SS_lib_type RF --left ALL.LEFT.fq --right ALL.RIGHT.fq --CPU 4 --JM 1G

Running Trinity on this data set may take 10 to 15 minutes. You'll see it progress through the various stages, starting with Jellyfish to generate the k-mer catalog, then followed by Inchworm, Chrysalis, and finally Butterfly.

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

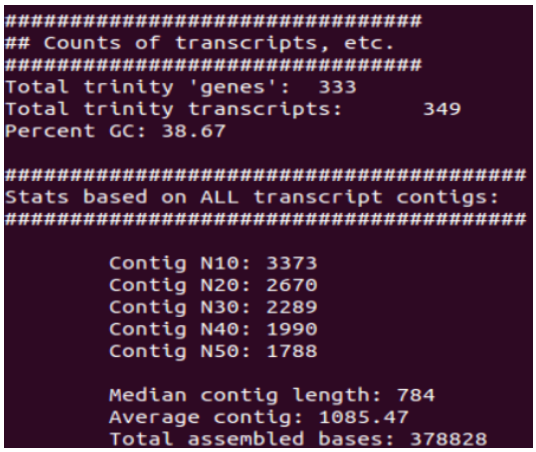
Just to look at the top few lines of the assembled transcript fasta file, you can run:

% 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



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 -n 0 -D . -d genome trinity_out_dir/Trinity.fasta -f samse > trinity_gmap.sam
```

(Note, you'll likely encounter warning messages such as "No paths found for comp42_c0_seq1", which just means that GMAP wasn't able to find a high-scoring alignment of that transcript to the targeted genome sequences.)

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:

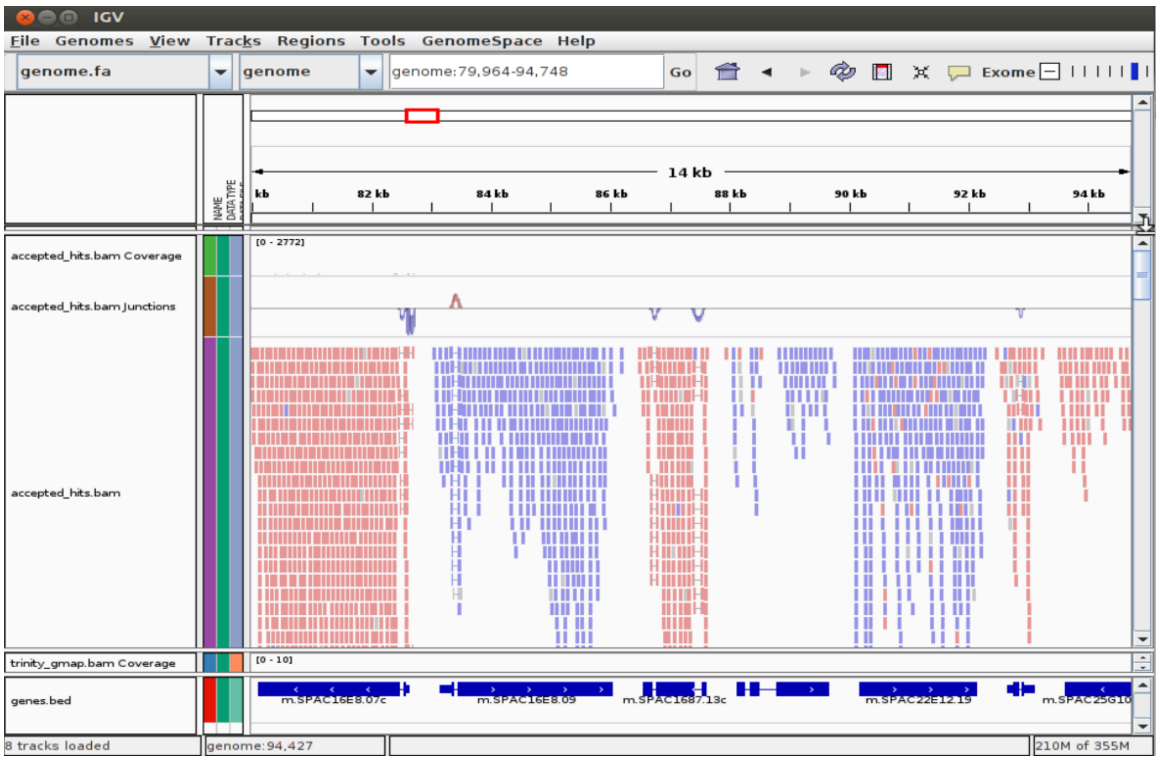
```
# prep the genome for running tophat
% bowtie-build genome.fa genome

# now run tophat:
% tophat -I 300 -i 20 --bowtie1 genome ALL.LEFT.fq ALL.RIGHT.fq

# 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 -Xmx2G -jar /home/ubuntu/software/IGV_2.3.12/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?

Exit the IGV viewer to continue on with the tutorial/demo.

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.

Separate transcript expression quantification for each of the samples:

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. This process is

% $TRINITY_HOME/util/align_and_estimate_abundance.pl --seqType fq --left Sp_ds.left.fq --right Sp_ds.right.fq --transcripts trinity_out_dir/Trinity.fasta --output_prefix Sp_ds --est_method RSEM --aln_method bowtie --trinity_mode --prep_reference

Once finished, RSEM will have generated two files: 'Sp_ds.isoforms.results' and 'Sp_ds.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.

Examine the format of the 'Sp_ds.isoforms.results' file by looking at the top few lines of the file:

% head Sp_ds.isoforms.results

transcript_id	gene_id	length	effective_length		expected_count	TPM	FPKM	IsoPct
c0_g1_i1	c0_g1	1768	1503.26	191.36	1507.01	1323.99	95.75	
c0_g1_i2	c0_g1	1973	1708.26	9.64	66.84	58.72	4.25	
c101_g1_i1	c101_g1	943	678.26	5.00	87.27	76.67	100.00	
c102_g1_i1	c102_g1	518	253.26	7.00	327.21	287.48	100.00	
c102_g2_i1	c102_g2	224	7.52	0.00	0.00	0.00	0.00	
c103_g1_i1	c103_g1	1639	1374.26	28.00	241.21	211.92	100.00	
c103_g2_i1	c103_g2	267	26.08	3.00	1361.89	1196.49	100.00	
c103_g3_i1	c103_g3	439	174.27	3.00	203.81	179.06	100.00	
c104_g1_i1	c104_g1	2463	2198.26	162.00	872.46	766.50	100.00	

Run RSEM on each of the remaining three samples:

% $TRINITY_HOME/util/align_and_estimate_abundance.pl --seqType fq --left Sp_hs.left.fq --right Sp_hs.right.fq --transcripts trinity_out_dir/Trinity.fasta --output_prefix Sp_hs --est_method RSEM --aln_method bowtie --trinity_mode --prep_reference

% $TRINITY_HOME/util/align_and_estimate_abundance.pl --seqType fq --left Sp_log.left.fq --right Sp_log.right.fq --transcripts trinity_out_dir/Trinity.fasta --

output_prefix Sp_log --est_method RSEM --aln_method bowtie --trinity_mode --prep_reference

% $TRINITY_HOME/util/align_and_estimate_abundance.pl --seqType fq --left Sp_plat.left.fq --right Sp_plat.right.fq --transcripts trinity_out_dir/Trinity.fasta --output_prefix Sp_plat --est_method RSEM --aln_method bowtie --trinity_mode --prep_reference.right.fq --transcripts trinity_out_dir/Trinity.fasta --prefix Sp_plat -- --no-bam-output

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/abundance_estimates_to_matrix.pl --est_method RSEM --out_prefix Trinity_trans Sp_ds.isoforms.results Sp_hs.isoforms.results Sp_log.isoforms.results Sp_plat.isoforms.results

later, we'll need the transcript length information, which we can extract from one of the RSEM.isoforms.results files like so:

% cat Sp_ds.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 Trinity_trans.counts.matrix --method edgeR

Examine the contents of the edgeR/ directory.

% ls edgeR/

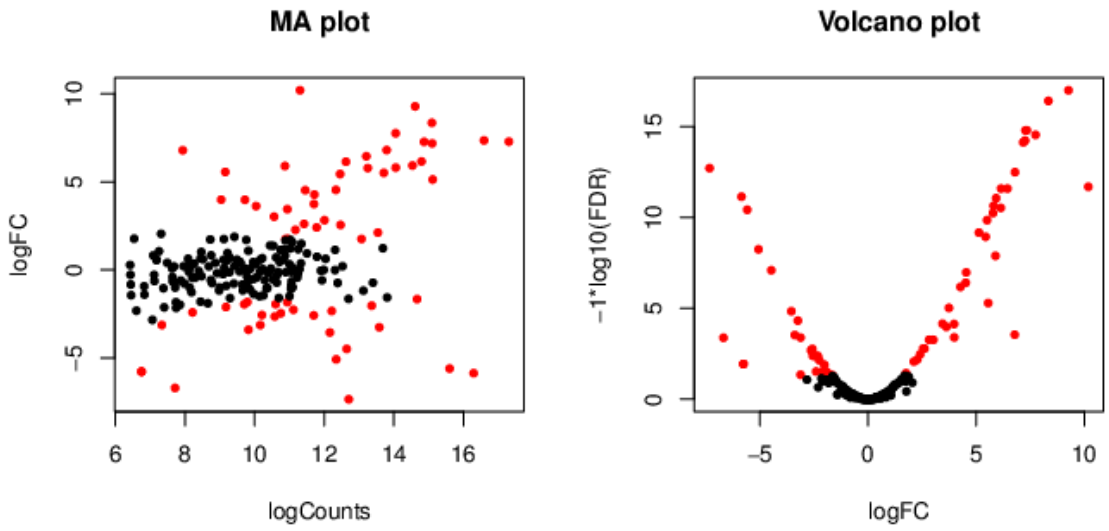
The files '*.DE_results' contain the output from running EdgeR to identify differentially expressed transcripts in each of the pairwise sample comparisons. Examine the format of one of the files, such as the results from comparing Sp_log to Sp_plat:

% head edgeR/Trinity_trans.counts.matrix.Sp_log_vs_Sp_plat.edgeR.DE_results

```
logFC   logCPM  PValue  FDR
c170_g1_i1      9.61861801691481        14.5952185340962        1.89965009068886e-20    5.54697826481149e-
18
c202_g1_i1      8.43972792091846        15.0838517672685        2.27480684206792e-19    3.32121798941917e-
17
c196_g1_i1      7.32063500194191        16.5748854452774        3.41413207352946e-17    3.08830499834592e-
15
c163_g1_i1      7.23640360489281        17.2895996370933        4.23055479225469e-17    3.08830499834592e-
15
c228_g1_i1      7.45783516784235        14.8586291284283        5.97264400230485e-17    3.48802409734603e-
15
c177_g1_i1      7.70480710969751        14.0382201624384        7.95551613891305e-17    3.87168452093769e-
15
c23_g1_i1       7.18728797492081        15.08171156382  2.20079814218603e-16    9.18047225026174e-15
c114_g1_i1      -7.33986527602987       12.7121596579517        7.37970295973721e-15    2.69359158030408e-
13
c119_g1_i1      6.75092725093354        13.7763384565024        1.63573118635626e-14    5.30703896017809e-
13
```

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:



Exit the chart viewer to continue.

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' edgeR/Trinity_trans.counts.matrix.Sp_log_vs_Sp_plat.edgeR.DE_results
| awk '{ if ($5 <= 0.05) print;}' | wc -l

    62
```

Trinity facilitates analysis of these data, including scripts for 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 Trinity_trans.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 'Trinity_trans.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 (DE) transcripts that are at least 4-fold differentially expressed at a significance of <= 0.001 in any of the pairwise sample comparisons:

```
% $TRINITY_HOME/Analysis/DifferentialExpression/analyze_diff_expr.pl --matrix
Trinity_trans.counts.matrix.TMM_normalized.FPKM –P 1e-3 –C2
```

The above generates several output files with a prefix “diffExpr.P0.001_C2”, indicating the parameters chosen for filtering, where P (FDR actually) is set to 0.001, and fold change (C) is set to 2^(2) or 4-fold. *(These are default parameters for the above script. See script usage before applying to your data).*

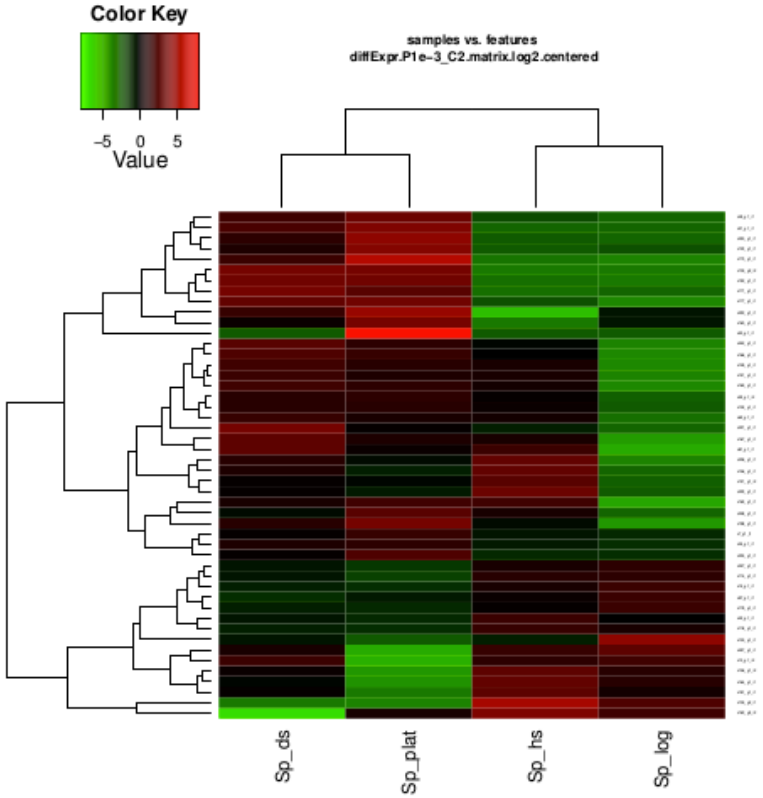
Included among these files are:
‘diffExpr.P0.001_C2.matrix’ : the subset of the FPKM matrix corresponding to the DE transcripts identified at this threshold. The number of DE transcripts identified at the specified thresholds can be obtained by examining the number of lines in this file.

% wc -l diffExpr.P1e-3_C2.matrix

49

Note, the number of lines in this file includes the top line with column names, so there are actually 48 DE genes at this 4-fold and 1e-3 FDR threshold cutoff.

Also included among these files is a heatmap ‘diffExpr.P1e-3_C2.matrix.heatmap.pdf’ as shown below, with transcripts clustered along the vertical axis and samples clustered along the horizontal axis.



Exit the PDF viewer to continue.

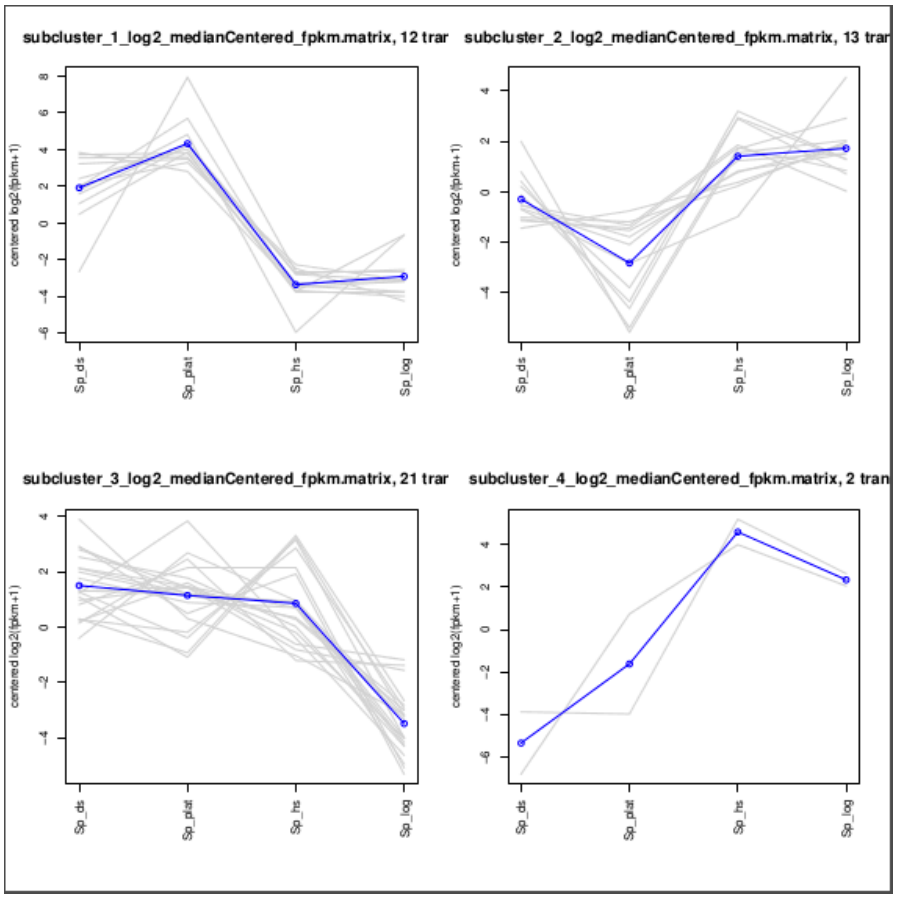
Extract transcript clusters by expression profile by cutting the dendrogram

Extract clusters of transcripts with similar expression profiles by cutting the transcript cluster dendrogram at a given percent of its height (ex. 60%), like so:

% $TRINITY_HOME/
Analysis/DifferentialExpression/define_clusters_by_cutting_tree.pl --Ptree 60 -R
diffExpr.P1e-3_C2.matrix.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.P1e-3_C2.matrix.RData.clusters_fixed_P_60/my_cluster_plots.pdf



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