Trinity De novo Transcriptome Assembly Workshop Report:

The report outlines a comprehensive workshop on Trinity De novo Transcriptome Assembly, covering the process of setting up the environment using Anaconda and performing RNA-Seq data analysis using Trinity and EdgeR. The workshop focused on Schizosaccharomyces pombe RNA-Seq data, involving different growth phases and stress conditions.

**Setting up the Environment with Anaconda:** Anaconda, a versatile package and environment management system, was used in the project. It provides a streamlined process for creating isolated environments with all necessary dependencies, making Trinity accessible across various operating systems, including Linux, macOS, and Windows. The environment was installed in a Linux operating system within a virtual machine.

**Downloading Data:** The workshop required strand-specific RNA-Seq data from Schizosaccharomyces pombe under four conditions: logarithmic growth (log), plateau phase (plat), diauxic shift (ds), and heat shock (hs). Each condition had 1 million paired-end reads, totaling 4 million paired-end reads. The data was obtained either through a web URL or the 'wget' command from the command line.

%wget [http://sourceforge.net/projects/trinityrnaseq/files/misc/TrinityNatureProtocolTutorial.tgz/](http://sourceforge.net/projects/trinityrnaseq/files/misc/TrinityNatureProtocolTutorial.tgz/download)

The downloaded file will be named 'TrinityNatureProtocolTutorial.tgz' and has a size of 540 MB. To extract the contents of the file, use the following command:

tar -xvf TrinityNatureProtocolTutorial.tgz

which should generate the following files in a TrinityNatureProtocolTutorial/ directory with

the following contents:

S\_pombe\_refTrans.fasta # reference transcriptome for S. pombe

1M\_READS\_sample/Sp.hs.1M.left.fq # PE reads for heatshock

1M\_READS\_sample/Sp.hs.1M.right.fq

1M\_READS\_sample/Sp.log.1M.left.fq # PE reads for log phase

1M\_READS\_sample/Sp.log.1M.right.fq

1M\_READS\_sample/Sp.ds.1M.right.fq # PE reads for diauxic shock

1M\_READS\_sample/Sp.ds.1M.left.fq

1M\_READS\_sample/Sp.plat.1M.left.fq # PE reads for plateau phase

1M\_READS\_sample/Sp.plat.1M.right.fq

samples\_n\_reads\_described.txt # tab-delimited description file.

A screen shot of a computer program

Description automatically generated

The reference data we used provided in the attached folder

**De novo Transcriptome Assembly:** Trinity software was utilized to perform de novo assembly using the raw RNA-Seq reads. Trinity's algorithms assembled reads into contigs, identified splice junctions, and generated a set of transcripts without relying on a reference genome. The resulting Trinity assembly was used for subsequent analyses.

**Inspecting the Assembly with a Reference Genome:** If a reference genome was available, the Trinity assembly was compared and analyzed within the context of the reference genome. Alignment tools like Bowtie, BWA, or STAR were used to align Trinity transcripts to the reference genome, allowing identification of their positions.

**Mapping Reads and Trinity Transcripts to a Target Genome:** If a target genome sequence different from the reference genome was available, reads and Trinity transcripts were mapped to this target genome. This step provided insights into the alignment and positioning of RNA-Seq reads and Trinity transcripts specific to the target genome.

**Visualizing Aligned Reads and Transcripts:** After aligning reads and Trinity transcripts to the reference or target genome, visualization tools like Integrative Genomics Viewer (IGV) or Genome Browse were employed. These tools allowed researchers to explore alignment results, compare them to known reference transcript annotations, and assess assembly accuracy and quality.

**Abundance Estimation using RSEM:** RSEM software was utilized for abundance estimation of transcripts reconstructed by Trinity. This involved aligning the original RNA-Seq reads back to the Trinity transcripts using Bowtie and estimating the number of RNA-Seq fragments mapping to each contig. Sample-specific abundance values were obtained, as the abundance of individual transcripts can vary significantly between samples.

**Differential Expression Analysis using EdgeR:** To detect differentially expressed transcripts, the Bioconductor package EdgeR was run using the counts matrix. The analysis identified transcripts with significant differential expression at a false discovery rate (FDR) of 0.05. The results included log fold change, log counts per million, p-values, and FDR values. MA and Volcano plots were generated to visualize the significant changes in expression.

**Extracting Differentially Expressed Transcripts:** The analysis extracted differentially expressed transcripts that were at least 4-fold differentially expressed with a significance of 0.001 or lower in any of the pairwise sample comparisons.

**Extracting Transcript Clusters by Expression Profile:** Transcript clusters with similar expression profiles were extracted by cutting the transcript cluster dendrogram at a specified percentage of its height (e.g., 60%). Individual transcript clusters were obtained and summarized with charts.

Overall, the workshop demonstrated the step-by-step process of Trinity De novo Transcriptome Assembly and subsequent analysis using EdgeR, enabling researchers to identify differentially expressed transcripts and gain valuable insights into gene expression changes across different conditions.

Methods and Results:

1. Generating Trinity De novo RNA-Seq Assembly: The workshop started with performing de novo assembly using the Trinity software. The raw RNA-Seq reads were used to reconstruct RNA transcripts without prior knowledge of a reference genome. Trinity utilized algorithms to assemble reads into contigs, identify splice junctions, and generate a set of transcripts.
2. Top of Form
3. Bottom of Form

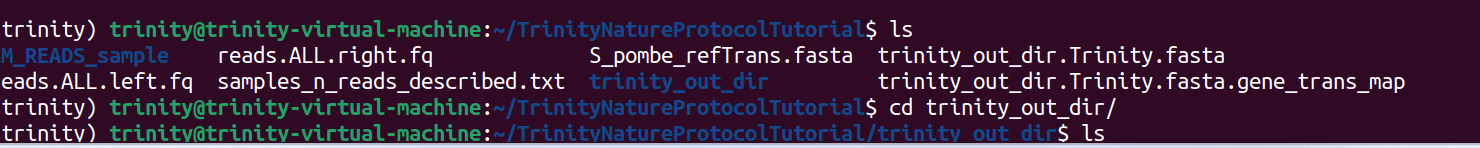
In order to facilitate downstream analyses, concatenate the RNA-Seq data across all

samples into a single set of inputs to generate a single reference Trinity assembly. Combine

all ‘left’ reads into a single file, and combine all ‘right’ reads into a single file by:

% cat 1M\_READS\_sample/\*.left.fq > reads.ALL.left.fq

% cat 1M\_READS\_sample/\*.right.fq > reads.ALL.right.fq



**De novo assembly of reads using Trinity:**

$Trinity --seqType fq --SS\_lib\_type RF \ --left reads.ALL.left.fq --right reads.ALL.right.fq --CPU 6 --max\_memory 10G

A computer screen shot of a computer code

Description automatically generated

This command progress through the various stages, starting with Jellyfish to generate the k-mer catalog, then followed by Inchworm, Chrysalis, and finally Butterfly.

The --JM option allows the user to control the amount of RAM used during Jellyfish kmer counting, in this case, 8 Gb of RAM. The --CPU option controls the number of parallel processes. Feel free to change these depending on your system.The assembled transcripts will be found at ‘trinity\_out\_dir.Trinity.fasta’.

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

%head trinity\_out\_dir.Trinity.fastaA screenshot of a computer screen

Description automatically generated

## Examine assembly stats:

The script 'TrinityStats.pl' provides information about the number of transcripts, components, and the transcript contig N50 value based on the 'Trinity.fasta' file. The contig N50 value is a metric commonly used to assess the contiguity of a genome assembly. It represents the maximum length at which at least 50% of the total assembled sequence is contained within contigs of that length.

It is important to note that while maximizing N50 is suitable for genome assemblies, it may not be the most appropriate approach for transcriptomes. Instead, it is recommended to use an index based on a reference dataset from the same or closely related species. This allows for estimation of the number of recovered reference genes and determination of how many can be considered full-length.

Nevertheless, the N50 value can still be useful for confirming the success of the assembly process. In the case of S. pombe, you would expect an N50 value that is close to the average transcript length of 1311 bases. To obtain this statistic for the Trinity assemblies, you can utilize the script:

% $TRINITY\_HOME/util/TrinityStats.pl trinity\_out\_dir.Trinity.fasta

In our device the trinity\_home is /home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/util/TrinityStats.pl

The command becomes:

% /home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/util/TrinityStats.pl trinity\_out\_dir.Trinity.fasta

A screenshot of a computer

Description automatically generated

## Compare de novo reconstructed transcripts to reference annotations:

As we have access to a reference genome and a set of reference transcript annotations corresponding to our dataset, we can align the Trinity contigs to the genome and examine them within the genomic context.

To begin, we need to prepare the genomic region for alignment using GMAP. Execute the following command to build the necessary index for alignment:

% gmap\_build -d genome -D . -k 13 GENOME\_data/genome.fa:

A computer screen with white text

Description automatically generated

This command generates the index required for aligning the Trinity transcript contigs to the genome.

Next, align the Trinity transcript contigs to the genome using GMAP. The alignment output will be in the SAM format, which simplifies data viewing in our genome browser. Execute the following command

%gmap -n 0 -D . -d genome trinity\_out\_dir.Trinity.fasta -f samse > trinity\_gmap.sam

A screen shot of a computer

Description automatically generated

Please note that you may encounter warning messages such as "No paths found for comp42\_c0\_seq1," indicating that GMAP was unable to find a high-scoring alignment for that specific transcript to the targeted genome sequences.

To facilitate further analysis and visualization, convert the alignment file to a coordinate-sorted BAM (binary SAM) format. Use the following command:

% samtools view -Sb trinity\_gmap.sam > trinity\_gmap.bam

This command converts the SAM file to BAM format.

Additionally, it is crucial to coordinate-sort the BAM file for compatibility with various tools, particularly viewers. Sort the BAM file with the following command:

% samtools sort trinity\_gmap.bam -o sorted\_trinity\_gmap.bam

Finally, to enable rapid navigation within the genome browser, index the sorted BAM file:

% samtools index trinity\_gmap.bam

By following these steps, we can align the Trinity contigs to the reference genome, generate a sorted BAM file, and create an index for efficient navigation in the genome browser.

A computer screen with text on it

Description automatically generated

### b. Align RNA-seq reads to the genome using hisat2

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

. % hisat2-build GENOME\_data/genome.fa genome\_index

A screenshot of a computer

Description automatically generated

%hisat2 -x genome\_index -1 reads.ALL.left.fq -2 reads.ALL.right.fq -I 300 --minins 20 --maxins 1000 -S alignment.sam

A screenshot of a computer

Description automatically generated

Abundance estimation using RSEM

To estimate the expression levels of the transcripts reconstructed by Trinity, we employ the RSEM software for abundance estimation. The approach involves aligning the original RNA-Seq reads back to the Trinity transcripts and then utilizing RSEM to estimate the number of RNA-Seq fragments that map to each contig. It is important to analyze the reads from each sample separately to obtain sample-specific abundance values since the abundance of individual transcripts can vary significantly between samples.

For the alignment step, we utilize 'bowtie' instead of 'tophat' for two reasons. Firstly, as we are mapping reads to reconstructed cDNAs rather than genomic sequences, there is no need for reads to be aligned across introns. Secondly, RSEM software is currently only compatible with gap-free alignments

To facilitate the usage of RSEM within the Trinity framework, the Trinity software includes scripts that wrap the RSEM software.

Next, we perform separate quantification of transcript expression for each sample using the following script. This script runs RSEM, which aligns the RNA-Seq reads to the Trinity transcripts using the Bowtie aligner and subsequently performs abundance estimation. The process is as follows:

%/home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/util/align\_and\_estimate\_abundance.pl --seqType fq --left 1M\_READS\_sample/Sp.ds.1M.left.fq --right 1M\_READS\_sample/Sp.ds.1M.right.fq --transcripts trinity\_out\_dir.Trinity.fasta --output\_prefix Sp\_ds --est\_method RSEM --aln\_method bowtie --prep\_reference --output\_dir Sp\_ds.RSEM

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 head Sp\_ds.RSEM/Sp\_ds.isoforms.results:

A screenshot of a computer screen

Description automatically generated

Run RSEM on each of the remaining three samples and examine their outputs:

Quantify expression for Sp\_h:

/home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/util/align\_and\_estimate\_abundance.pl --seqType fq --left 1M\_READS\_sample/Sp.hs.1M.left.fq --right 1M\_READS\_sample/Sp.hs.1M.right.fq --transcripts trinity\_out\_dir.Trinity.fasta --output\_prefix Sp\_hs --est\_method RSEM --aln\_method bowtie --prep\_reference --output\_dir Sp\_hs.RSEM

A screenshot of a computer screen

Description automatically generated

Quantify expression for Sp\_plat:

/home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/util/align\_and\_estimate\_abundance.pl --seqType fq --left 1M\_READS\_sample/Sp.plat.1M.left.fq --right 1M\_READS\_sample/Sp.plat.1M.right.fq --transcripts trinity\_out\_dir.Trinity.fasta --output\_prefix Sp\_plat --est\_method RSEM --aln\_method bowtie --prep\_reference --output\_dir Sp\_plat.RSEM

A screenshot of a computer screen

Description automatically generated

Quantify expression for Sp\_log:

(trinity) trinity@trinity-virtual-machine: ~/TrinityNatureProtocolTutorial$ /home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/util/align\_and\_estimate\_abundance.pl --seqType fq --left 1M\_READS\_sample/Sp.log.1M.left.fq --right 1M\_READS\_sample/Sp.log.1M.right.fq --transcripts trinity\_out\_dir.Trinity.fasta --output\_prefix Sp\_log --est\_method RSEM --aln\_method bowtie --prep\_reference --output\_dir Sp\_log.RSEM

A screenshot of a computer screen

Description automatically generated

Generate a counts matrix and perform cross-sample normalization:

(trinity) trinity@trinity-virtual-machine:~/TrinityNatureProtocolTutorial$ /home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/util/abundance\_estimates\_to\_matrix.pl --est\_method RSEM \

--out\_prefix Trinity\_trans \

Sp\_ds.RSEM/Sp\_ds.isoforms.results \

Sp\_hs.RSEM/Sp\_hs.isoforms.results \

Sp\_log.RSEM/Sp\_log.isoforms.results \

Sp\_plat.RSEM/Sp\_plat.isoforms.results

A screenshot of a computer program

Description automatically generated

**Differential Expression Using EdgeR**

To detect differentially expressed transcripts, run the Bioconductor package edgeR using our counts matrix:

%/home/trinity/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/Analysis/DifferentialExpression/run\_DE\_analysis.pl --matrix ~/TrinityNatureProtocolTutorial/Trinity\_trans.counts.matrix --method edgeR --dispersion 0.1 --output edgeR

% ls -ltr edgeR/

A screenshot of a computer screen

Description automatically generated

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:

A computer screen with white text

Description automatically generated

The data obtained from the EdgeR analysis includes log fold change (logFC), log counts per million (logCPM), p-value from an exact test, and false discovery rate (FDR).

Based on these data, the EdgeR analysis generated both MA and Volcano plots. You can examine any of the 'edgeR/transcripts.counts.matrix.condA\_vs\_condB.edgeR.DE\_results.MA\_n\_Volcano.pdf' PDF files to explore these plots

A volcano plot is a graphical representation commonly used in genomics and bioinformatics to visualize the results of differential expression analysis or hypothesis testing for large-scale datasets. It provides a quick way to identify genes or features that are significantly differentially expressed between two or more experimental conditions. The plot is called a "volcano plot" due to its characteristic shape, which resembles the silhouette of a volcano.

Here's how to interpret a volcano plot:

1. **X-axis**: The X-axis represents the effect size or fold change of each gene or feature between the compared conditions. It is typically presented on a logarithmic scale. Positive values on the right side indicate upregulation (higher expression in condition 2 compared to condition 1), while negative values on the left side indicate downregulation (higher expression in condition 1 compared to condition 2).
2. **Y-axis**: The Y-axis represents the statistical significance or p-values associated with each gene or feature. The p-value measures the probability of observing the differential expression by chance alone. Lower p-values indicate higher statistical significance, meaning that the gene is less likely to be differentially expressed due to random variation.
3. **Data Points**: Each gene or feature is represented as a data point on the plot. The position of the data point is determined by its fold change (X-axis) and its corresponding p-value (Y-axis).
4. **Significantly Differentially Expressed Genes**: Genes that are highly differentially expressed and statistically significant will appear far away from the center of the plot, either towards the top-right (upregulated) or bottom-left (downregulated). These are often the most biologically relevant findings, as they indicate substantial changes in expression that are unlikely to occur by chance.
5. **Non-Significantly Differentially Expressed Genes**: Genes with fold changes close to 1 (near the center of the plot) are typically considered non-differentially expressed. These genes may show small changes in expression that are not statistically significant.
6. **Threshold Lines**: Often, researchers draw vertical and horizontal lines on the plot to indicate certain significance and fold-change thresholds. For example, a common practice is to set a p-value threshold (e.g., p < 0.05) for statistical significance and a fold-change threshold (e.g., log2 fold change > 1 or < -1) to identify the most differentially expressed genes.

volcano plot provides a visually intuitive way to identify genes that are both significantly differentially expressed and biologically relevant in a high-dimensional dataset. Researchers use this plot to quickly identify potential candidate genes for further investigation or to gain insights into the underlying biology of the experimental conditions being studied.

The MA plot is a graphical representation widely used in genomics and bioinformatics to visualize the results of high-throughput experiments, such as microarray or RNA-Seq data. The "MA" stands for "M" (log-ratio) and "A" (average expression), and the plot helps identify genes or features that are differentially expressed between two or more conditions.

Here's how to define and interpret the MA plot:

1. **M (Log-Ratio)**: The M value represents the log-ratio of the expression levels between two conditions. It is calculated as the logarithm (usually base 2) of the fold change in expression between the two conditions. Mathematically, M is calculated as follows:

M = log2(Expression in Condition B) - log2(Expression in Condition A)

The M value indicates the magnitude of change in expression. If a gene has a positive M value, it means it is upregulated in Condition B compared to Condition A. If the M value is negative, the gene is downregulated in Condition B compared to Condition A. Larger M values indicate more significant fold changes in expression.

1. **A (Average Expression)**: The A value represents the average expression level of a gene across the two conditions. It is calculated as the average of the log-transformed expression values for the two conditions. Mathematically, A is calculated as follows:

A = 0.5 \* (log2(Expression in Condition A) + log2(Expression in Condition B))

The A value represents the magnitude of expression for a gene, regardless of whether it is differentially expressed or not.

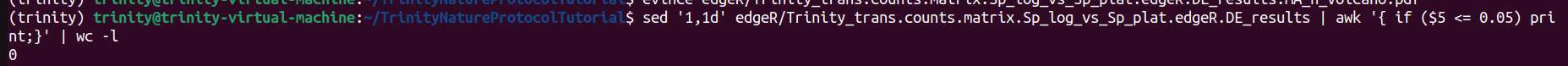
1. **Data Points**: Each gene or feature is represented as a data point on the MA plot, with its M value on the Y-axis and its A value on the X-axis.
2. **Interpretation**:
   * Differentially Expressed Genes: Genes with significant changes in expression between the two conditions will appear as data points far away from the center line (A=0). The further a data point is from the center line, the more significant the change in expression. Upregulated genes will have positive M values, and downregulated genes will have negative M values.
   * Non-Differentially Expressed Genes: Genes with little or no change in expression between the two conditions will appear near the center line (A=0), where the M value is close to zero.
3. **Threshold Lines**: Researchers often add horizontal lines on the MA plot to indicate certain thresholds for statistical significance and fold change. This can help identify genes that meet predefined criteria for differential expression.

The MA plot is a powerful and intuitive tool to visualize differential expression data. It allows researchers to quickly identify genes with significant changes in expression and can provide valuable insights into the underlying biology of the experimental conditions being studied.

A screenshot of a computer screen

Description automatically generated

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



Now let's perform the following operations from within the edgeR/ directory. Enter the edgeR/ dir like so:

% cd edgeR/

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:

~/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/Analysis/DifferentialExpression/analyze\_diff\_expr.pl --matrix ../Trinity\_trans.TMM.EXPR.matrix -P 1e-3 -C 2

The above generates several output files with a prefix “diffExpr.P1e-3\_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.P1e-3\_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.



Note, the number of lines in this file includes the top line with column names, so there are actually 637 DE transcripts at this 4-fold and 1e-3 FDR threshold cutoff.

Also included among these files is a heatmap ‘diffExpr.P1e-3\_C2.matrix.log2.centered.genes\_vs\_samples\_heatmap.pdf’ as shown below, with transcripts clustered along the vertical axis and samples clustered along the horizontal axis.

A screenshot of a computer

Description automatically generated

## 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:

% ~/anaconda3/pkgs/trinity-2.1.1-6/opt/trinity-2.1.1/Analysis/DifferentialExpression/define\_clusters\_by\_cutting\_tree.pl \

--Ptree 20 -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:

% evince diffExpr.P1e-3\_C2.matrix.RData.clusters\_fixed\_P\_60/my\_cluster\_plots.pdf

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

Challenges:

1. Outdated Paper and Command Updates: The paper used for the Trinity tutorial was outdated, and many of the commands in it had been updated or changed, leading to confusion and difficulties in replicating the procedures.
2. Debugging Errors: During the tutorial, numerous errors were encountered, requiring extensive debugging to identify and rectify issues in the analysis pipeline.
3. Platform Compatibility: The tutorial was not compatible with Windows, necessitating the installation of Linux, which added complexity and additional steps for Windows users.
4. Software Dependencies: The Trinity tutorial had specific software versions and dependencies that needed to be installed and configured correctly. This posed a challenge, particularly for users new to bioinformatics and command-line tools.
5. Overcoming Dependency Issues with Conda: To address the dependency challenges, we utilized Conda, a package manager, to manage and resolve the various software dependencies, simplifying the installation process.
6. Interpretation of Results: Upon completing the analysis, the next challenge was interpreting the results and deriving meaningful biological insights from the vast amount of generated data. This required expertise in transcriptomics and gene expression analysis to make sense of the findings.