Supplementary Materials for De-novo transcript sequence reconstruction from RNA-Seq: reference generation and analysis with Trinity

Sections:

S1. Evaluation of transcriptome assembly completeness

A challenge in *de novo* sequencing assembly is determining whether the given sequencing depth is sufficient to meet a project's goals. The difference in transcript coverage by RNA-Seq reads due to the large dynamic range transcripts' expression makes it difficult to determine what is the required depth of sequencing. In many cases, the required sequencing depth depends on the sensitivity needed to detect, assemble, and/or reliably quantify lowly expressed transcripts.

Saturation plots, which examine assembly-related quantities as a function of sequencing depth, can provide insights into the effects of sequencing coverage on transcriptome assembly and allow one to infer the impacts of deeper sequencing. **Supplementary Fig. 6 and 7** illustrate the effects of sequencing depth on the cumulative assembly size, numbers of Trinity contigs and components generated, and the numbers of genes found with full-length reconstructed transcripts for *S. pombe* and mouse transcriptomes, respectively. While some features approach saturation, others continue to grow with increased depth and are less useful for assessing completeness. For example, the total sum of gene lengths (CSum_E100; computed by summing the expected fragment count from the corresponding RSEM 'gene.results' files; **Supplementary Fig. 6b and 7b**) does not show evidence of saturation. This is largely due to comparably large sums of gene lengths that correspond to the weakliest supported transcripts having the fewest reads mapped (right-most data points in **Supplementary Fig. 6a and 7a**). In contrast, the sum of component lengths representing the top 85% of mapped reads (CSum_E85) does saturate. In fact, similar CSum_E-statistics at or below 99% of expressed reads appear to provide useful indicators as to whether sufficient sequencing depth has been obtained.

Similarly, the total number of Trinity transcripts and components ('genes') does not saturate, despite a clear inflection point where sequencing depth begins to allow reads to aggregate into transcripts and genes. Thus, the number of transcripts and components reported increases at a substantial rate with further sequencing (**Supplementary Fig. 6d and 7d**). However, the number of *full-length* transcripts reconstructed, as compared to the reference transcripts for each organism (performed as in [1]) shows evidence of saturation.

Such direct analyses of full-length transcript reconstruction using reference genome-based transcript sets are impossible in the context of most applications of *de novo* transcriptome assemblies, as the targets are typically not model organisms and lack genome sequences with high quality annotations. Instead, an analogous analysis can be performed based on identifying the number of homologous proteins found to be nearly fully represented by contiguous alignments to *de novo* reconstructed transcripts. To this end, The Trinity software package includes a script

```
$TRINITY_HOME/util/analyze_blastPlus_topHit_coverage.pl
```

to assist in identifying BLAST matches to known proteins and to bin them according to coverage of the homologous protein's length (see **Tutorial**). By searching Trinity's transcripts against the Swissprot proteins (having first removed mouse and *Schizosaccharomyces* proteins), we identified all homologous proteins having a best match to a Trinity transcript and aligning to at least 80% of the homologous protein's length. Similarly to the analysis of the full-length reference transcripts using reference genome annotations, we find that the number of full-length homologous protein matches saturates. Analyzing length coverage of homologous proteins is thus an excellent indicator of the impact of sequencing depth on the resulting transcriptome assembly, and provides a metric for assembly quality.

S2. Comparison of expression analyses between Trinity *de novo* assemblies and a reference transcriptome

To assess the utility of Trinity *de novo* transcriptome assemblies for analyzing transcript expression and for identifying differentially expressed transcripts across multiple biological conditions, we conducted a series of experiments to compare Trinity assemblies to a reference transcriptome – that of *Schizosaccharomyces pombe* (fission yeast), using previously published RNA-Seq data[2]. We selected five million paired-end strand-specific RNA-Seq fragment reads generated from *S. pombe* under four different growth states: logarithmic growth, plateau phase, heat shock, and diauxic shift. These reads are available at: http://sourceforge.net/projects/trinityrnaseq/files/misc/SP2.DiffExpr.5Mea_fqs.tgz/download

The reads from each of the growth conditions were combined into a single data set containing 20M paired-end reads and assembled into transcripts using Trinity by:

```
Trinity.pl --seqType fq --JM 20G --left SP2.all.5M.LEFT.fq --right SP2.all.5M.RIGHT.fq \
        --CPU 5 --SS_lib_type RF
```

The resulting Trinity-assembled transcripts were then mapped to the reference transcriptome of *S. pombe* using BLASTN, followed by comparisons of Trinity assemblies to the reference transcripts. This assumes that the expression values and identified differentially expressed transcripts identified using the reference transcriptome are a gold standard for assessing accuracy.

Reference transcripts (5,064) were extracted from the *S. pombe* genome based on the protein-coding gene annotations provided at GeneDB (as previously described in [2]), including an additional 100 bases at each end to ensure that paired reads extending into UTR regions would be captured effectively. Antisense transcript sequences were generated for each of the reference sense transcripts by reverse-complementing the sense transcript sequences. Reciprocal best BLASTN matches between the reference transcripts and Trinity assemblies were identified, comparing only the top strand of each nucleotide sequence in order to maintain strand-specificity in the mappings. When a Trinity assembly was derived from a component containing multiple assembled transcripts (e.g. multiple transcript isoforms), we further isolated those reciprocal best matches whereby each trinity assembly for that component had a best top BLASTN match to a consistent reference transcript, identifying 4,306 Trinity components and reference transcript pairs, excluding antisense mappings.

RNA-Seq reads from each growth condition were separately aligned to the reference transcripts and to all Trinity-assembled transcripts, and abundance estimates were computed for each condition using RSEM[3]. We also computed Trinity component-level abundances (analogous to gene-level expression values) by summing RSEM-estimated abundances for all transcripts generated from an individual component.

The average transcript sequence length per Trinity component was compared to the reference transcript sequence length, and the percentage of the shorter sequence length was computed and binned at 10% intervals. The expression values for each Trinity component and each of the four conditions were analyzed according to each percent length bin, and Pearson correlation values were computed to compare the expression values between reference transcripts and the mapped Trinity components for each condition. The expression values appear to be highly correlated and the correlation tends to improve as transcripts are more completely reconstructed (**Supplementary Fig. 8**).

EdgeR[4] was used to identify differentially expressed transcripts as described in the main text and tutorial. EdgeR was applied separately to the reference transcripts and to the Trinity components, identifying 471 reference transcripts and 469 Trinity components at least 4-fold differentially expressed with an FDR-corrected P-value of 0.001, containing an intersection of 430 equivocal (reciprocally linked) entries. The expression profiles across the four growth conditions were found to be nearly identical between the reference transcripts and corresponding Trinity components, with correlations between growth conditions ranging from 0.95 to 0.99 (**Supplementary Fig. 9**).

S3. TransDecoder Algorithm

TransDecoder was developed to identify likely protein-coding regions within transcript sequences. The algorithm is as follows.

1. All open reading frames (ORFs) above a minimum length (default: 100 amino acids) are identified across all transcripts. ORFs require a Methionine start codon and an in-frame stop codon, unless the ORF is found at either terminus of the transcript, allowing for partial ORFs. If the transcripts are identified to be strand-specific, then only the top strand is examined for ORFs.
2. The top 500 longest ORFs (configurable parameter) are selected and a reading frame-specific 5th-order Markov model is trained based on these coding sequences.
3. All previously identified ORFs are scored as a sum of the log odds ratio across the length of the putative coding sequence, similarly to [5]. The log odds score *L* at a given nucleotide position *i* of the coding sequence is computed as:

   $$L_i = \log \frac{p(c_i^F \mid c_{i\_1 \dots i\_5})}{p(c_i^B)}$$

   where *F* is the reading frame position {1,2,3} of nucleotide *c* at position *i* of the putative coding sequence. $p(c_i^B)$ corresponds to the relative frequency of nucleotide c within the entire set of transcript sequences.
   The complete ORF is scored by summing the per base log-likelihood scores across the length of the sequence.
4. Each ORF is scored according to its putative reading frame and compared to its score as computed for each of the five alternative reading frames. If the score in the first reading frame is positive and exceeds the scores for each of the five alternative (and presumably incorrect) reading frames, then it is reported as a candidate likely coding region.

In addition to reporting ORFs meeting the above requirements, any ORF found to exceed a minimum length of 900 bases (encoding a peptide of at least 300 amino acids) (configurable parameter) is reported. The user also has the option to include a search of Pfam, and any ORFs found to match a Pfam domain with a score meeting the noise cutoff will additionally be included as a candidate. A future version of TransDecoder is planned to include an optional BLAST search against a protein database as a way of further retaining candidates for further study.

TransDecoder is included within the Trinity software package and is separately maintained at SourceForge at http://transdecoder.sf.net. This tool is first reported here.

S4. *In silico* normalization of RNA-Seq fragments

Assembling large numbers of reads (many hundreds of millions of RNA-Seq fragments) can be computationally intensive and lead to extensive runtimes, although most transcripts are likely to have been fully saturated by read coverage at much lower sequencing depth. Hence, deeper coverage may not benefit the assembly of these already saturated transcripts and only lead to an overall increase in computational burden. To assemble RNA-Seq data more efficienty using Trinity, reads may first be normalized according to depth of sequencing coverage using tools included in the Trinity software distribution, specifically by running:

```
$TRINITY_HOME/util/normalize_by_kmer_coverage.pl
```

The required parameters for the above script are similar to those of Trinity.pl (eg. –left, --right, --single, --JM, --seqType), with the addition of '–max_cov' that should be set to the maximum targeted coverage. We recommend this value to be set to at least 30, indicating a targeted maximum coverage of at most 30X.

The *in silico* read normalization algorithm is largely based on that described for the 'diginorm' software[6] but with modifications as described below. Similarly to diginorm, a catalog of k-mers from all the reads is obtained and the median k-mer abundance (C) is treated as a proxy for the abundance of that sequencing read, or RNA-Seq fragment in the case of paired reads. However, instead of discarding reads having median k-mer abundances exceeding the targeted coverage (T), each read is retained with probability min(1, T/C), which both captures all reads falling below the targeted coverage level and down-samples those reads occurring at higher coverage to the targeted coverage level. In addition, the distribution of k-mer coverage across the length of each read is analysed and those reads showing an aberrant k-mer abundance profile, defined as having a standard deviation in k-mer coverage that exceeds the median k-mer abundance value, are rejected.

The *in silico* normalization step is implemented in three phases. First, Jellyfish[7] is used to build a catolog of all k-mers (default set as k=25), starting from FASTQ or FASTA input files containing the RNA-Seq data (as in the current version of Trinity). Second, the k-mers are parsed and stored in a hash table along with abundance values. RNA-Seq reads are parsed and the k-mer abundance profile is computed by looking up the abundance value for each k-mer encountered across the length of the read. The median, mean, and standard deviation for the k-mer coverage is reported for each read, and written to a file (with .stats extension). In the case of paired fragment reads, the left and right reads are processed separately, which can be done in parallel (if parameter –PARALLEL_STATS is invoked). Also, in the case of paired reads, the left and right fragment reads can be normalized separately or together as individual fragments, where both the left and right fragment reads are altogether retained or discarded based on the filtering criteria. To filter paired reads as individual fragments, invoke the '—pairs_together' parameter. This will average the k-mer abundance statistics between the left and right reads, generating a 'pairs.stats' file.

Third, the relevant .stats file(s) are parsed and reads (or whole fragments) are selected probabilistically based on the median k-mer coverage, or immediately excluded as aberrant based on the criteria described above. The output of the *in silico* normalization is a file(s) containing a proper subset of the input read data. These output FASTQ or FASTA files are then used as regular inputs to Trinity. We should note that, although the normalized reads are well suited to Trinity assembly, the original (un-normalized) reads should be used for downstream operations involving transcript abundance estimation.

S5. Advanced Butterfly parameterization

Butterfly contains a number of parameters that effect transcript reconstruction, from the initial de Bruijn graph pruning and compaction, determining whether sufficient read support exists to support transcript extension, and for determining whether alternate path sequences are sufficiently different to report as distinct transcript isoforms. **Supplementary Fig. 1-5** highlight some of the most important parameters one might tune based on characteristics of their RNA-Seq data. Each advanced parameter setting is listed below and described in the legend accompanying the corresponding figure panel, including anticipated impacts from adjusting their values:

- Graph compaction parameters (**Supplementary Fig. 1**).

  --edge-thr

  --flow-thr

- Transcript path extension read (pair) overlap requirements (**Supplementary Fig. 2**).

  --path_reinforcement_distance

  -R

- Merging insufficiently different path sequences during reconstruction (**Supplementary Fig. 3**).

  –min_per_id_same_path

  –max_diffs_same_path

  –max_internal_gap_same_path

- Reducing combinatorial path construction via triplet-locking (**Supplementary Fig. 4**):

  –triplet-lock (Butterfly) or –no_triplet_lock (Trinity.pl)

- Reducing combinatorial path construction by path restriction (**Supplementary Fig. 5**):

  –max_number_of_paths_per_node

S6. Timing of execution of the Tutorial

Hardware used:

- Model: PowerEdge R815
- RAM = 512GB
- CPU = AMD 2.2 G
- Physical = 4
- Cores = 48

Timing:

# Preparing reads

```
TIME: 0.1 min. for cat 1M_READS_sample/Sp.ds.1M.left.fq >> reads.ALL.left.fq
TIME: 0.1 min. for cat 1M_READS_sample/Sp.ds.1M.right.fq >> reads.ALL.right.fq
TIME: 0.1 min. for cat 1M_READS_sample/Sp.hs.1M.left.fq >> reads.ALL.left.fq
TIME: 0.1 min. for cat 1M_READS_sample/Sp.hs.1M.right.fq >> reads.ALL.right.fq
TIME: 0.1 min. for cat 1M_READS_sample/Sp.log.1M.left.fq >> reads.ALL.left.fq
TIME: 0.1 min. for cat 1M_READS_sample/Sp.log.1M.right.fq >> reads.ALL.right.fq
TIME: 0.1 min. for cat 1M_READS_sample/Sp.plat.1M.left.fq >> reads.ALL.left.fq
TIME: 0.1 min. for cat 1M_READS_sample/Sp.plat.1M.right.fq >> reads.ALL.right.fq
```

# Running Trinity

```
TIME: 71.8 min. for $TRINITY_HOME/Trinity.pl --left reads.ALL.left.fq --right
reads.ALL.right.fq  --seqType fq  --JM 10G  --CPU 6  --SS_lib_type RF
```

# Abundance estimation using RSEM

```
TIME: 8.5 min. for $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl  --
transcripts trinity_out_dir/Trinity.fasta  --seqType fq  --prefix ds_rep1  --left
1M_READS_sample/Sp.ds.1M.left.fq --right 1M_READS_sample/Sp.ds.1M.right.fq  --
SS_lib_type RF  --thread_count 6
```

```
TIME: 0.0 min. for cat ds_rep1.isoforms.results | cut -f1,3,4 > Trinity.trans_lengths.txt
```

```
TIME: 9.2 min. for $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl  --
transcripts trinity_out_dir/Trinity.fasta  --seqType fq  --prefix hs_rep1  --left
1M_READS_sample/Sp.hs.1M.left.fq --right 1M_READS_sample/Sp.hs.1M.right.fq  --
SS_lib_type RF  --thread_count 6
```

```
TIME: 8.4 min. for $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl  --
transcripts trinity_out_dir/Trinity.fasta  --seqType fq  --prefix log_rep1  --left
1M_READS_sample/Sp.log.1M.left.fq --right 1M_READS_sample/Sp.log.1M.right.fq  --
SS_lib_type RF  --thread_count 6
```

```
TIME: 8.2 min. for $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl  --
transcripts trinity_out_dir/Trinity.fasta  --seqType fq  --prefix plat_rep1  --left
```

1M_READS_sample/Sp.plat.1M.left.fq --right 1M_READS_sample/Sp.plat.1M.right.fq --SS_lib_type RF --thread_count 6

# Generating the count matrices for DE analysis

TIME: 0.0 min. for $TRINITY_HOME/util/RSEM_util/merge_RSEM_frag_counts_single_table.pl ds_rep1.isoforms.results hs_rep1.isoforms.results log_rep1.isoforms.results plat_rep1.isoforms.results > Trinity_trans.counts.matrix

# TMM normalization and writing the FPKM matrices

TIME: 0.1 min. for $TRINITY_HOME/Analysis/DifferentialExpression/run_TMM_normalization_write_FPKM_matrix.pl --matrix Trinity_trans.counts.matrix --lengths Trinity.trans_lengths.txt

# DE analysis using edgeR

TIME: 0.3 min. for $TRINITY_HOME/Analysis/DifferentialExpression/run_DE_analysis.pl --matrix Trinity_trans.counts.matrix --method edgeR --samples_file samples_n_reads_desribed.txt --output edgeR_trans

TIME: 2.0 min. for $TRINITY_HOME/Analysis/DifferentialExpression/analyze_diff_expr.pl --matrix ../Trinity_trans.counts.matrix.TMM_normalized.FPKM

References

1. Grabherr, M.G. et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature biotechnology* **29**, 644-652 (2011).
2. Rhind, N. et al. Comparative functional genomics of the fission yeasts. *Science* **332**, 930-936 (2011).
3. Li, B. & Dewey, C.N. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC bioinformatics* **12**, 323 (2011).
4. Robinson, M.D., McCarthy, D.J. & Smyth, G.K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26**, 139-140 (2010).
5. Parra, G., Blanco, E. & Guigo, R. GeneID in Drosophila. *Genome research* **10**, 511-515 (2000).
6. Brown, C.T., Howe, A., Zhang, Q., Pryrkosz, A.B. & Brom, T.H. A Reference-Free Algorithm for Computational Normalization of Shotgun Sequencing Data. *arXiv:1203.4802 [q-bio.GN]* (2012).
7. Marcais, G. & Kingsford, C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* **27**, 764-770 (2011).