PROTOCOL

De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis

Brian J Haas[1,21], Alexie Papanicolaou[2,21], Moran Yassour[1,3], Manfred Grabherr[4], Philip D Blood[5], Joshua Bowden[6], Matthew Brian Couger[7], David Eccles[8], Bo Li[9], Matthias Lieber[10], Matthew D MacManes[11], Michael Ott[2], Joshua Orvis[12], Nathalie Pochet[1,13], Francesco Strozzi[14], Nathan Weeks[15], Rick Westerman[16], Thomas William[17], Colin N Dewey[9,18], Robert Henschel[19], Richard D LeDuc[19], Nir Friedman[3] & Aviv Regev[1,20]

[1]Broad Institute of Massachusetts Institute of Technology (MIT) and Harvard, Cambridge, Massachusetts, USA. [2]Commonwealth Scientific and Industrial Research Organisation (CSIRO) Ecosystem Sciences, Black Mountain Laboratories, Canberra, Australian Capital Territory, Australia. [3]The Selim and Rachel Benin School of Computer Science, The Hebrew University of Jerusalem, Jerusalem, Israel. [4]Science for Life Laboratory, Department of Medical Biochemistry and Microbiology, Uppsala University, Uppsala, Sweden. [5]Pittsburgh Supercomputing Center, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. [6]CSIRO Information Management & Technology, St. Lucia, Queensland, Australia. [7]Department of Microbiology and Molecular Genetics, Oklahoma State University, Stillwater, Oklahoma, USA. [8]Genomics Research Centre, Griffith University, Gold Coast Campus, Gold Coast, Queensland, Australia. [9]Department of Computer Sciences, University of Wisconsin, Madison, Wisconsin, USA. [10]Center for Information Services and High-performance Computing (ZIH), Technische Universität Dresden, Dresden, Germany. [11]California Institute for Quantitative Biosciences, University of California, Berkeley, Berkeley, California, USA. [12]Institute for Genome Sciences, Baltimore, Maryland, USA. [13]Department of Plant Systems Biology, Vlaams Instituut voor Biotechnologie (VIB), Department of Plant Biotechnology and Bioinformatics, Ghent University, Ghent, Belgium. [14]Parco Tecnologico Padano, Località Cascina Codazza, Lodi, Italy. [15]Corn Insects and Crop Genetics Research Unit, United States Department of Agriculture–Agricultural Research Service, Ames, Iowa, USA. [16]Genomics facility, Purdue University, West Lafayette, Indiana, USA. [17]GWT-TUD GmbH, Saxony, Germany. [18]Department of Biostatistics and Medical Informatics, University of Wisconsin, Madison, Wisconsin, USA. [19]University Information Technology Services, Research Technologies Division, Indiana University, Bloomington, Indiana, USA. [20]Department of Biology, Howard Hughes Medical Institute, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. [21]These authors contributed equally to this work. Correspondence should be addressed to B.J.H. (bhaas@broadinstitute.org) or A.R. (aregev@broad.mit.edu).

Published online 11 July 2013; doi:10.1038/nprot.2013.084

***De novo* assembly of RNA-seq data enables researchers to study transcriptomes without the need for a genome sequence; this approach can be usefully applied, for instance, in research on 'non-model organisms' of ecological and evolutionary importance, cancer samples or the microbiome. In this protocol we describe the use of the Trinity platform for *de novo* transcriptome assembly from RNA-seq data in non-model organisms. We also present Trinity-supported companion utilities for downstream applications, including RSEM for transcript abundance estimation, R/Bioconductor packages for identifying differentially expressed transcripts across samples and approaches to identify protein-coding genes. In the procedure, we provide a workflow for genome-independent transcriptome analysis leveraging the Trinity platform. The software, documentation and demonstrations are freely available from http://trinityrnaseq.sourceforge.net. The run time of this protocol is highly dependent on the size and complexity of data to be analyzed. The example data set analyzed in the procedure detailed herein can be processed in less than 5 h.**

INTRODUCTION

High-throughput sequencing of genomes (DNA-seq) and transcriptomes (RNA-seq) has opened the way to study the genetic and functional information stored within any organism at an unprecedented scale and speed. For example, RNA-seq in principle enables the simultaneous study of transcript structure (such as alternative splicing), allelic information (e.g., SNPs) and expression with high resolution and broad dynamic range[1]. These advances greatly facilitate functional genomics research in species for which genetic or financial resources are limited, including many non-model organisms—organisms that, although they have not been extensively studied in a research setting, are nevertheless of substantial ecological or evolutionary importance.

Although many genomic applications have traditionally relied on the availability of high-quality genome sequences, such sequences have only been determined for a very small portion of known organisms. Furthermore, sequencing and assembling a genome is still a costly endeavor in many cases, owing to genome size and repeat content. Conversely, as only a fraction of the genome is transcribed, RNA-seq data can provide a rapid and cheaper 'fast track'—within reach of any lab—to delineating a reference transcriptome for downstream applications, such as alignment, phylogenetics or marker construction. Indeed, even within a whole-genome sequencing project, RNA-seq has become an essential source of evidence for the identification of transcribed genes and the annotation of exon structure.

Realizing the full potential of RNA-seq requires computational methods that can assemble a transcriptome even when a genome sequence is not available. Two primary methods exist for converting raw RNA-seq data into transcript sequences: through the guidance of assembled genomic sequences or via *de novo* assembly[2,3]. The genome-guided approach to transcriptome studies has quickly become a standard approach to RNA-seq analysis for model organisms, and several software packages exist for this purpose[4,5]. This approach cannot, however, be applied to organisms for which a well-assembled genome does not exist, and even for organisms having well-assembled genomes the results may vary across genome assembly versions. In such cases, a *de novo* transcriptome assembler is required. However, the process of assembling a transcriptome violates many of the assumptions on which assemblers developed for application on genomic DNA data rely. For example, uniform coverage and the 'one locus-one contig' paradigm are not valid for RNA: an accurate transcriptome assembler will produce one contig per distinct transcript (isoform) rather than per locus, and different transcripts will have different coverage, reflecting their different expression levels.

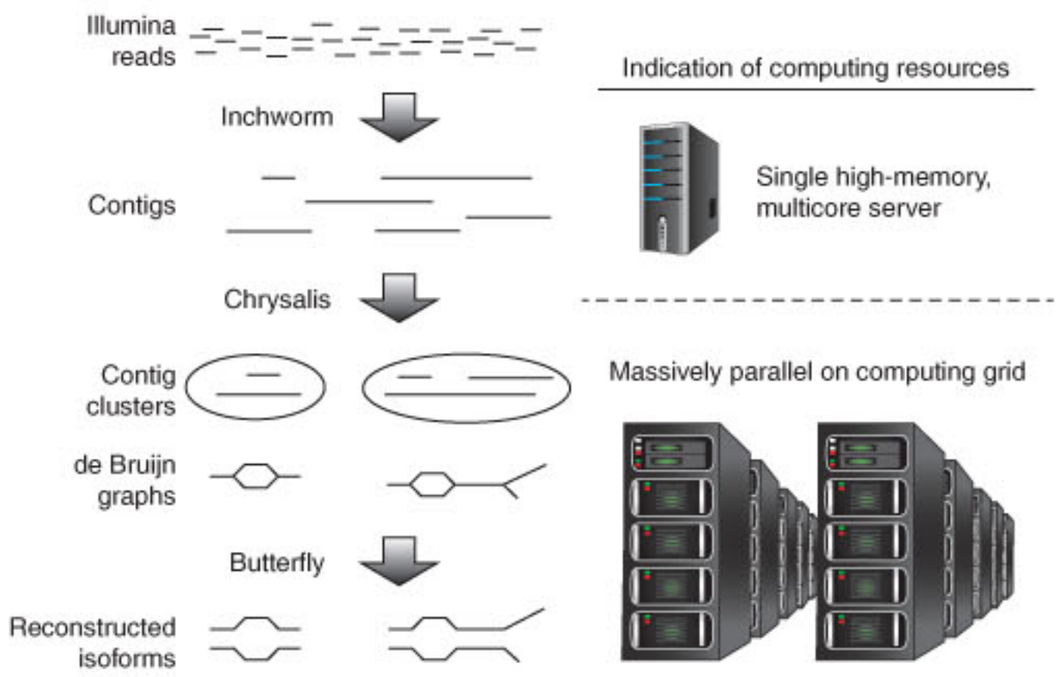

Figure 1 | Overview of Trinity assembly and analysis pipeline. Key sequential stages in Trinity (left) and the associated computational resources (right). Trinity takes as input short reads (top left) and first uses the Inchworm module to construct contigs. This stage requires a single high-memory server (~1 GB of RAM per 1 million paired reads, but varies according to read complexity; top right). Chrysalis (middle left) clusters related Inchworm contigs, often generating tens to hundreds of thousands of Inchworm contig clusters, each of which is processed to a de Bruijn graph component independently and in parallel on a computing grid (bottom right). Butterfly (bottom left) then extracts all probable sequences from each graph component, which can be parallelized as well.

Several tools are now available for *de novo* assembly of RNA-seq. Trans-ABySS[6], Velvet-Oases[7] and SOAPdenovo-trans (http://soap.genomics.org.cn/SOAPdenovo-Trans.html) are all extensions of earlier developed genome assemblers. We previously described a novel alternative method for transcriptome assembly called Trinity[8]. Trinity partitions RNA-seq data into many independent de Bruijn graphs (ideally one graph per expressed gene), and it uses parallel computing to reconstruct transcripts from these graphs, including alternatively spliced isoforms. Trinity can leverage strand-specific Illumina paired-end libraries, but it can also accommodate non-strand-specific and single-end-read data. Trinity reconstructs transcripts accurately with a simple and intuitive interface that requires little to no parameter tuning. Several independent studies have demonstrated that Trinity is highly effective compared with alternative methods (e.g., refs. 9–11; The DREAM Project's Alternative Splicing Challenge, http://www.the-dream-project.org/result/alternative-splicing). The high number of citations that Grabherr *et al.*[8] has accrued in a relatively short time (since its online publication in May 2011) further corroborates Trinity's utility. Trinity users study a broad range of model and non-model organisms from all kingdoms, and they come from small laboratories and large genome projects alike (e.g., the pea aphid genome annotation v2; Fabrice Legeai (Institut National de la Recherche Agronomique (INRA)) and Terence Murphy (RefSeq National Center for Biotechnology Information (NCBI)), personal communications to A. P.).

Trinity also has an active developer community, which has greatly enhanced its performance and utility (see http://trinityrnaseq.sourceforge.net). For example, although the run-time performance of the first release was not computationally efficient[11], the Trinity developer community has since improved its efficiency, halving memory requirements and increasing processing speed through increased parallelization and improved algorithms (Henschel *et al.*[12] and M.O., unpublished data). Furthermore, Trinity was converted into a modular platform that seamlessly uses third-party tools, such as Jellyfish[13], to build the initial *k*-mer catalog. Other third-party tools integrated into Trinity have enhanced the utility of its reconstructed transcriptomes. For example, Trinity now supports tools (e.g., RSEM[14], edgeR[15] and DESeq[16]) that take its output transcripts and test for differential expression while accounting for both technical and biological sources of variation[17–19] and correcting for multiple hypothesis testing. Given Trinity's popularity and substantial enhancements since publication, it is important to provide detailed procedures that leverage its various features. The procedures we present here will further expand Trinity's utility for studies in non-model organisms.

Overview of the Trinity RNA-seq assembler

Trinity's assembly pipeline consists of three consecutive modules: Inchworm, Chrysalis and Butterfly (**Fig. 1**). We strongly encourage users to first read Trinity's first publication[8] for an extensive description of the method, which we present here more briefly.

First, all overlapping *k*-mers are extracted from the RNA-seq reads. Inchworm then examines each unique *k*-mer in the decreasing order of abundance, and it generates transcript contigs using a greedy extension based on (*k*-1)-mer overlaps. Inchworm often generates full-length transcripts for a dominant isoform, but it reports just the unique portions of alternatively spliced transcripts. This method of operation works well for data sets that are largely deficient in repetitive sequences, such as transcriptomes.

Next, Chrysalis first clusters related Inchworm contigs into components, using the raw reads to group transcripts on the basis of shared read support and paired read links, when available. This process clusters together regions that have probably originated from alternatively spliced transcripts or closely related gene families. Chrysalis then encodes the structural complexity of clustered Inchworm contigs by building a de Bruijn graph for each cluster, and it then partitions the reads among the clusters. The partitioning of the Inchworm contigs and RNA-seq reads into disjointed clusters ('components') enables massively parallel processing of subsequent computations.

Finally, Butterfly processes the individual graphs in parallel, ultimately reporting full-length transcripts for alternatively spliced isoforms and teasing apart transcripts that correspond to paralogous genes. Butterfly traces the RNA-seq reads through the graph and determines connectivity based on the read sequence and on further support from any available paired-end data. When connections cannot be verified by traced reads, Butterfly will split the graph into several disconnected sub-graphs and process each separately. Butterfly then traverses the supported graph paths and reconstructs transcript sequences in a manner that reflects the original cDNA molecules.

We describe key issues related to Trinity's operation in **Boxes 1–4** (see also **Figs. 2–6** and **Supplementary Figs. 1–5**), including the following: requirements of the input sequence data and the optional use of *in silico* normalization to reduce the quantity of the input reads to be assembled and to improve assembly efficiency (**Box 1**); computing requirements and the availability of computing resources to users for running Trinity (**Box 2**); the basics of

Box 1 | Input sequence data requirements for assembly

This protocol requires users to supply short-read data in either FASTQ or FASTA formats. Although these reads may be either paired-end or single-end, paired-end sequence data are preferred, as they are able to guide more distant connections between regions of transcript isoforms during assembly. Strand-specific transcript sequencing is also highly recommended so that sense and antisense transcription can be readily distinguished. If multiple sequencing runs are conducted for a single experiment, these reads may be concatenated into a single-read file for single-end sequencing, or into two files (e.g., merging all 'left' and all 'right' reads into single 'left.fq' and 'right.fq' files, respectively) in the case of paired-end sequencing. Similarly, if multiple biological or technical replicates are sequenced, these data can be concatenated into individual files. Trinity may be used with data of any read length commonly produced by next-generation sequencers, but most of our experience stems from the use of either 76- or 101-base Illumina reads.

For paired-end reads, Trinity must identify those reads that correspond to opposite ends of a sequenced molecule. Trinity requires reads in either FASTQ or FASTA format to have read names (accessions) that include a /1 or /2 suffix to indicate the left or right end of the sequenced fragment. For example:

(first entry of the 'left.fq' file)

```
@61DFRAAXX100204:1:100:10494:3070/1
ACTGCATCCTGGAAAGAATCAATGGTGGCCGGAAAGTGTTTTTCAAATACAAGAGTGACAATGTGCCCTGTTGTTT
+
ACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBC?CCCCCCCCC@@CACCCCCACCCCCCCCCCCCCCCCCCCCCC
```

(first entry of the 'right.fq' file)

```
@61DFRAAXX100204:1:100:10494:3070/2
CTCAAATGGTTAATTCTCAGGCTGCAAATATTCGTTCAGGATGGAAGAACATTTTCTCAGTATTCCATCTAGCTGC
+
C<CCCCCCCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBCCCCCCCCCCCCCCCCACCCCCACCC=
```

If the Casava 1.8 format for FASTQ is used (see below), Trinity will reconstruct the read name from

```
@HWI-ST896:156:D0JFYACXX:5:1101:1652:2132 1:N:0:GATCAG
```

to

```
HWI-ST896:156:D0JFYACXX:5:1101:1652:2132/1
```

Pre-processing sequence data. Although preassembly sequence quality control steps are not required, performing a few easy steps can improve performance. First, if the reads include barcodes used for multiplexing on the sequencing instrument, all barcodes must be removed before running Trinity, for example, by running Trimmomatic[44] or cutadapt[45].

Second, removing reads (or regions of reads) that probably include sequencing errors may reduce the complexity of the resulting de Bruijn graph and hence improve RAM usage and program runtime. Specifically, the program Trimmomatic[44] can successfully remove terminal nucleotides characterized by a quality that is lower than a user-supplied minimum threshold (e.g., Q15).

Third, if more than 200 million paired-end sequences are to be assembled, the user may consider performing an *in silico* normalization of the sequencing reads. Deep RNA-seq produces vast numbers of reads for transcripts that are already well represented at lower sequencing depths in addition to providing increased sensitivity for the detection of rarer transcripts[36,46]. Normalization decreases runtime by reducing the total number of reads, while largely maintaining transcriptome complexity and capability for full-length transcript reconstruction[47]. Trinity includes an *in silico* read normalization utility inspired by the algorithm described for Diginorm[47], where each RNA-seq fragment (single read or pair of reads) is probabilistically selected based on its median *k*-mer coverage value and the targeted maximum coverage value (**Supplementary Note**).

On analyzing RNA-seq data sets from fission yeast[8,41] and mouse[8], we have found that normalization to as low as 30× *k*-mer ($k = 25$) coverage (23–31% of the reads) results in full-length reconstruction to an extent approaching that based on the entire read set, and far exceeds the full-length reconstruction performance, when simply subsampling the same number of reads from the total data set (**Fig. 2**). Although *in silico* normalization can better enable Trinity assembly of large RNA-seq data sets, and it is clearly better than the alternative of subsampling reads, the sensitivity for full-length transcript reconstruction may be affected, such as leading to the 6% decrease in full-length transcript reconstruction observed for the 30× maximum coverage normalization obtained on the basis of our mouse RNA-seq data (**Fig. 2**). However, the relative impact on the percentage of alternatively spliced reference transcript isoforms detected remains largely unchanged.

running Trinity (**Box 3**); and advanced operations, such as leveraging strand-specific RNA-seq (**Box 4**). Additional issues relevant to evaluating *de novo* transcriptome assemblies, including examining the completeness of an assembly and estimating the potential impact of deeper sequencing, are addressed in the **Supplementary Note** and in **Supplementary Figures 6** and **7**.

Transcriptome analysis package for non-model organisms

Generating a *de novo* RNA-seq assembly is only the first step toward transcriptome analysis. Common goals for studying transcriptomes in both model and non-model organisms include identifying transcripts, characterizing transcript structural complexity and coding content, and understanding which genes and isoforms are

Box 2 | Computational requirements

The Trinity software is designed for Unix-type operating systems (primarily Linux); it provides a command-line interface and is best run on a high-memory, multicore computer or in a high-performance computing environment. In general, we recommend having ~1 GB of RAM per 1 million paired-end reads. A typical configuration is a multicore server with 256 GB to 1 TB of RAM, and such systems have become more affordable in the recent years ($15,000–40,000; markedly less expensive than many high-performance instruments used in molecular biology, and probably within reach of a departmental core facility). Smaller data sets can be processed in computing environments with reduced memory resources (e.g., see the PROCEDURE, which can be completed on a laptop with 8 GB of RAM). For research groups that lack the required computing resources, such resources are freely accessible to eligible researchers via the Data-Intensive Academic Grid (DIAG, http://diagcomputing.org), and services are available to researchers in the United States (and their international collaborators) through XSEDE, on systems such as 'Blacklight' at the Pittsburgh Supercomputing Center (http://www.psc.edu/index.php/trinity).

As Trinity is executed from command line, users should have a basic familiarity with operating in a Unix environment. Each of Trinity's three core modules has a different characteristic run time, memory usage and parallelization (**Fig. 1**). Although the entire Trinity computing pipeline can be executed on a single high-memory machine, the later stages, including Chrysalis 'QuantifyGraph' section and the Butterfly computations, benefit from access to a compute farm, where they can be massively parallelized. To this end, Trinity's final massively parallel section integrates the ability to submit to Load Sharing Facility, a grid-scheduling system. This objective is achieved through the use of the command-line parameter '--grid_computing_module' identifying a user-defined module for submitting commands to the grid for execution. A submission script ('trinity_pbs.sh') is also provided for using Trinity with the Portable Batch System Torque and PBSpro cluster job schedulers.

expressed in different samples (tissues, environmental conditions and so on). Trinity supports the achievement of these goals leveraging additional popular software already likely to be installed in a bioinformatics environment, by incorporating additional open-source software as plug-in components that are directly included in the Trinity software suite, and by providing easy-to-use scripts that aim to provide a familiar and friendly command-line interface to otherwise complex analysis modules. Results are often provided as tab-delimited files that users can import into their favorite spreadsheet programs, and visualizations are generated in PDF format. Below we describe the details of the individual protocol steps and identify the currently supported software utilities.

Comparing transcriptomes across samples

In many cases, a user will wish to compare the type or level of transcripts between samples, for example, for differentially expressed genes. Two possible routes exist to achieve this goal. One option is to assemble reads corresponding to each of the sample types separately and then compare the results from each of the assemblies. This approach, however, is complicated by the need to match the 'same' transcripts derived from the independent assemblies. A more straightforward alternative, and the one that we recommend implementing, is to first combine all reads across all samples and biological replicates into a single RNA-seq data set, assemble the reads to generate a single reference Trinity assembly (**Fig. 7**), and then quantify the level of each of these transcripts in each sample by aligning each sample's (not normalized) reads to the reference transcriptome assembly and counting the number of reads that align to each transcript (oversimplified here, detailed below). Finally, statistical tests are applied to compare the counts of reads observed for each transcript across the different samples, and those transcripts observed to have significantly different representation by reads across samples are reported. Further analysis of the differentially expressed transcripts can reveal patterns of gene expression and yield insights into relationships among the investigated samples.

Transcript abundance estimation

Transcript quantification is a prerequisite for many downstream investigations. Several metrics have been proposed for measuring transcript abundance levels based on RNA-seq data, normalizing for depth of sequencing and the length of transcripts. These metrics include reads per kilobase of target transcript length per million reads mapped (RPKM[20]) for single-end sequences, and an analogous computation based on counting whole fragments (FPKM[21]) for paired-end RNA-seq data.

To calculate the number of RNA-seq reads or fragments that were derived from transcripts, the reads must first be aligned to the transcripts. When you are working with a reference genome and an annotated transcriptome, reads are usually aligned to one or both[4]. In a *de novo* assembly setting, the reads are re-aligned to the assembled transcripts. However, alternatively spliced isoforms and recently duplicated genes may share sub-sequences that are longer than a single read or read pair, and these reads will map equally well to multiple targets. Several methods[4,14,22] were recently developed to estimate how to correctly allocate such reads to transcripts in a way that best approximates the transcripts' true expression levels. Among these is the RSEM (RNA-seq by Expectation Maximization) software[14], which uses an iterative process to fractionally assign reads to each transcript based on the probabilities of the reads being derived from each transcript (**Fig. 8**), taking into account positional biases created by RNA-seq library-generating protocols.

RSEM comes bundled with the Trinity software distribution. The RSEM software currently requires gap-free alignments of RNA-seq reads to Trinity-reconstructed transcripts, such as alignments generated by the Bowtie software[23]. Given the Trinity-assembled transcripts and the RNA-seq reads generated from a sample, RSEM will directly execute Bowtie to align the reads to the Trinity transcripts and then compute transcript abundance, estimating the number of RNA-seq fragments corresponding to each Trinity transcript, including normalized expression values as FPKM. In addition to estimating the expression levels of individual transcripts, RSEM

Box 3 | Basic Trinity operation

The assembly pipeline for a typical Trinity assembly is executed from the command line via a single Perl script called 'Trinity.pl', with options describing the sequencing protocol and the names of the files containing RNA-seq reads. For the Unix-style commands shown throughout, we use the initial character '%' as a command prompt, and the `$TRINITY_HOME` variable corresponds to the location of the Trinity software installation directory. Commands that span multiple lines are separated by a trailing '\' character.
If the RNA-seq protocol used a non-strand-specific method, then Trinity.pl is executed as follows:

For single-end reads:

```
% $TRINITY_HOME/Trinity.pl --seqType fq \
  --single single.fq --JM 20G
```

For paired-end reads:

```
% $TRINITY_HOME/Trinity.pl --seqType fq \
  --left left.fq \
  --right right.fq --JM 20G
```

The --seqType parameter indicates the file format for the sequenced reads, which can be in FASTQ ('fq') or FASTA ('fa') format (see **Box 1** and the PROCEDURE for more detailed descriptions of input requirements). Trinity will also detect which flavor of FASTQ is used (Sanger, Illumina 1.3, or CASAVA 1.8+) and convert them to FASTA files; Trinity does not currently use the base quality scores provided in the FASTQ file format. If the data are paired, the new FASTA file will have /1 and /2 header information to indicate 'pairedness'. When other formats of FASTQ or if paired-end FASTA files are provided to Trinity, users must ensure that this pairedness is represented using the /1 and /2 read name suffix notation.

Users can include additional parameter settings (see below) to tune any of the three assembly steps according to the characteristics of the data set, but Trinity usually performs well with the default parameters. Furthermore, some settings, such as '--JM 20G' above (20 GB of memory to be allocated to Jellyfish for building the initial *k*-mer catalog), relate to the hardware being used, and **Box 2** describes how users can select optimal settings for different hardware configurations.

Trinity output. The final output from running Trinity is a single FASTA-formatted file containing the reconstructed transcript sequences, found as 'trinity_out_dir/Trinity.fasta'. To understand the output format, recall that Chrysalis divides sequences into graph components based on sub-sequences shared between them, and Butterfly further refines the sequence's classification by partitioning components into sets of transcript contigs based on read support within the graph (**Fig. 1**).
An example of a pair of entries found in the Trinity FASTA-formatted output is as follows (**Fig. 3**):

```
>comp0_c0_seq1 len=5528 path=[3647:0-3646 129:3647-3775 1752:3776-5527]
AATTGAATCCCTTTTTGTATTGAAAAAGTTGAAATGAAAGACATATACAGAT
TGAATGTG...TCCTCTGATACACAGCCTCGCAGGGTTCATTTCAAGCCGTGGG
GCTGCGCCACGGGTGCTAAGTCAACTGCATTCGATGCGGCTTTTAAACCCCC
AGGGGACACCTCGGCCAGCTGTTTGCCTGCAGTA...TTGTGTTTCTTCAACAG
TTTATCAGCTTGCTGAATTGCCATTTTATTATTTCCATTATCAAGATAATCG
TAAATGGGCCGGAGGCGCCGGTCGTTAGGGTCCTGCACATGGCCCCGCGTCG
CCATGATGACAAGCGCAGAACCTCAGT
>comp0_c0_seq2 len=5399 path=[3647:0-3646 1752:3647-5398]
AATTGAATCCCTTTTTGTATTGAAAAAGTTGAAATGAAAGACATATACAGAT
TGAATGTG.TTGTGTTTCTTCAACAGTTTATCAGCTTGCTGAATTGCCATT
TTATTATTTCCATTATCAAGATAATCGTAAATGGGCCGGAGGCGCCGGTCGT
TAGGGTCCTGCACATGGCCCCGCGTCGCCATGATGACAAGCGCAGAACCTCA
GT
```

The FASTA header describes how the transcript was structurally represented as reconstructed by Butterfly. For example, in the header of the first reported sequence, the accession value 'comp0_c0_seq1' corresponds to i) Chrysalis component 'comp0', ii) Butterfly disconnected subgraph 'c0', iii) Butterfly-reconstructed sequence 'seq1' and iv) having a length of 5,528 bases. The path of the sequence traversed by Butterfly through nodes in the sequence graph (**Fig. 3a**, blue, red and green nodes) is provided in the header as 'path=[3647:0–3646 129:3647–3775 1752:3776–5527]', listing the identifiers of the ordered nodes (3647, 129, 1752) and the ranges within the reconstructed transcript sequence that correspond to each respective node.

The second sequence above is a second transcript derived from the same component, identified by the accession 'comp0_c0_seq2' that corresponds to a sequence 'seq2' output from the same Chrysalis component and Butterfly subgraph 'comp0_c0'. In this case, the path traverses only two of the nodes 3647 and 1752 (**Fig. 3a**, blue and green nodes), through the edge connecting them directly. Thus, 'seq1' differs from 'seq2' only by the addition of the sequence in the internal node '129' (central sub-sequence shown in bold italics; **Fig. 3a**, red node). Such variations can result from alternative splicing. Here, for example, comparison with the reference mouse genome shows that the internally unique sequence in 'seq1' corresponds to a cassette exon that is skipped in 'seq2' (**Fig. 3c**, where Trinity 'seq1' and 'seq2' are shown as 'Isoform B' and 'Isoform A', respectively). Such transcripts can be validated by comparison with an annotated reference genome (even that of a related species) or experimentally. In some cases, alternative paths reflect the shared and distinct portions in paralogous genes or alternative alleles. Mate-pairing information and sufficiently long reads enable Trinity to resolve phased variations and correctly reconstruct the individual isoforms or paralogous transcripts from the more complex graphs[8].

computes 'gene-level' estimates using the Trinity component as a proxy for the gene. To compare expression levels of different transcripts or genes across samples, a Trinity-included script invokes edgeR to perform an additional TMM (trimmed mean of *M*-values) scaling normalization that aims to account for differences in total cellular RNA production across all samples[24,25].

Both full-length and partially reconstructed Trinity transcripts can be useful to estimate gene expression, as compared

Box 4 | Advanced Trinity operations

Leveraging strand-specific RNA-seq data. Trinity's preferred input is strand-specific RNA-seq data, which enables it to distinguish between sense and antisense transcripts and minimizes erroneous fusions between neighboring transcriptional units that are encoded on opposite strands. This characteristic of Trinity is particularly useful when the Trinity approach is applied to dense microbial genomes, in which overlapping transcriptional units are common. Several methods are available for generating strand-specific RNA-seq[48,49]. When given strand-specific data, Trinity first converts all the input reads to the transcribed strand orientation, reverse-complementing the strand if required. Users need to indicate strand specificity to Trinity.pl via the '--SS_lib_type' parameter, with 'F' or 'R' values for single-end reads to indicate reads originating from the transcribed or opposite strand, respectively. Similarly, 'FR' or 'RF' values are used for paired-end reads to reflect both ends (**Fig. 4**). For example, the dUTP-based strand-specific sequencing generates 'RF' paired-end sequences, where the right read (name ending with /2) corresponds to the transcribed strand, and the left read (name ending with /1) exists in the opposite strand. A corresponding Trinity assembly command would be constructed as follows:

For single-end reads:

```
% $TRINITY_HOME/Trinity.pl --seqType fq --single single.fq \
  --JM 20G --SS_lib_type F
```

For paired-end reads:

```
% $TRINITY_HOME/Trinity.pl --seqType fq --left left.fq \
  --right right.fq --JM 20G --SS_lib_type RF
```

During the FASTA conversion, Trinity reverse-complements the read sequences that are specified to exist in the 'R' orientation.

The use of strand-specific data can cause a small increase in running time, owing to the increased *k*-mer complexity of the data. Specifically, in strand-specific data, the forward and reverse-complemented *k*-mers are stored individually, whereas in non-strand-specific data there is no distinction between a *k*-mer in either orientation, and a *k*-mer is stored in a single canonical representation. However, this small run-time increase yields several benefits, including a small overall improvement in transcript reconstruction compared with non-strand-specific data (**Fig. 5**), a substantial reduction in the number of falsely fused transcripts in species with compact genomes, such as fission yeast or *D. melanogaster* (**Fig. 5**), and the ability to distinguish sense and antisense transcripts, thus revealing otherwise concealed mechanisms for transcriptional regulation[8,41].

Mitigating falsely fused transcripts. To further resolve erroneous fusion of overlapping transcripts from close neighboring genes, Trinity can leverage mate-pair information (with the '--jaccard_clip' parameter) to identify and dissect regions within assembled contigs that are consistent with overlapping yet distinct transcripts (**Fig. 6**). In our experiments, roughly half of fused transcripts in fission yeast and *D. melanogaster* are resolved using this method, albeit at the cost of doubling to tripling of the total runtime (**Fig. 5**). As this operation has little effect on the quality of transcriptomes reconstructed from gene-sparse genomes, such as that of many vertebrates, such as, for instance, the mouse (**Fig. 5**), users are encouraged to use '--jaccard_clip' in the case of transcriptome assemblies for organisms expected to have compact genomes, such as microbial eukaryotes, where erroneous fusions are more likely to be generated.

Additional parameters and approaches to consider. Several other options can be tuned to further improve accuracy and reduce runtime. First, even though Trinity handles substitution-type sequencing errors well, it cannot detect adapters or contaminants that survive the poly-dT enrichment (poly-A capture) protocols. In Trinity's assemblies, it is not uncommon to find sequences from other species (such as viruses and bacteria) or native nonpolyadenylated transcripts that are highly abundant (such as rRNA). Although some such sequences can yield important insights (e.g., viral genomes in tumors[50]), in some cases users will opt to prefilter them. In particular, read pre-processing, such as *in silico* normalization of read quantities, quality trimming or read filtering can reduce graph complexity and resulting runtimes (**Box 1**).

Second, when performing very deep sequencing (e.g., in order to identify rare transcripts), the larger number of observed sequencing errors contributes to increased graph complexity and longer runtimes. In most cases, setting the minimum *k*-mer coverage requirement to 2 instead of the default of 1 via the '--min_kmer_cov 2' parameter setting will effectively handle such cases, eliminating the singly occurring *k*-mers that are heavily enriched in sequencing errors, and thus vastly decreasing the complexity of the graph. In cases of very deep sequencing (beyond several hundred million paired-end reads), users can normalize their data (**Box 1**) and/or consider performing a Trinity assembly with '--min_kmer_cov 2', and then align the original read data set back to the final Trinity transcripts for abundance estimation.

Third, Trinity's final phase, Butterfly, operates in parallel on graphs from individual clusters and, by default, uses identical parameters for each cluster. As most clusters can be processed by Butterfly quite rapidly (from seconds to a few minutes), users who have domain-specific knowledge can explore a number of parameter settings relating to graph traversal to better understand how well assembled is a particular cluster of transcripts. Such parameter adjustments may include redefining the read overlap requirements for path extension, or tuning the minimum edge weight thresholds at branch points in the graph (details are provided in the **Supplementary Note** and in **Supplementary Figs. 1–5**).

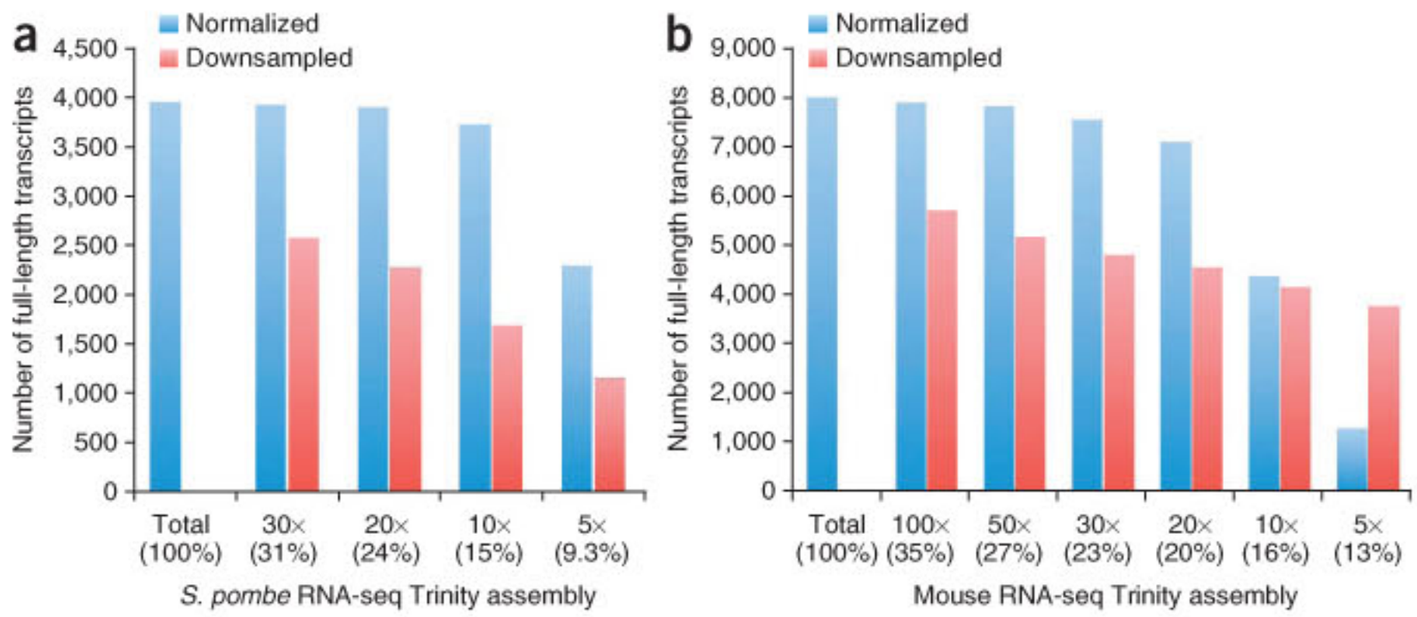

Figure 2 | Effects of *in silico* fragment normalization of RNA-seq data on Trinity full-length transcript reconstruction. (**a**,**b**) The *y* axis shows the number of full-length transcripts reconstructed from a data set of paired-end strand-specific RNA-seq in *S. pombe* (10 million paired-end reads) (**a**) and mouse (100 million, paired-end reads) (**b**), using either the full data set (total; 100%) or different samplings (*x* axis) by either Trinity's *in silico* normalization procedure (at 5× up to 100× targeted maximum *k*-mer ($k = 25$) coverage; blue bars)) or random downsampling of the same number of reads (red bars).

with estimating expression levels using a high-quality, reference genome–based transcript annotation. However, the more completely reconstructed transcripts tend to be more highly correlated with the expression levels estimated for reference transcripts (**Supplementary Note** and **Supplementary Fig. 8**).

Analysis of differentially expressed transcripts

To estimate differential gene expression between two types of samples, at least three biological replicates of each sample should ideally be obtained. Collection of replicate data enables to test whether the observed differences in expression are significantly different from expected biological variation under the null hypothesis that transcripts are not differentially expressed. In the absence of biological replicates, it is still possible to identify differentially expressed transcripts by using statistical models of expected variation, such as under the Poisson or negative binomial distribution. The Poisson distribution models well the variation expected between technical replicates[26], whereas the negative binomial distribution fares better in accounting for the increased variation observed between biological replicates, and it is a favored model for identifying differentially expressed transcripts by leading software tools[15,16].

Trinity transcriptome assemblies can serve as useful substrates for evaluating changes in gene expression between samples, and results are largely consistent with studies based on reference transcriptomes (**Supplementary Note** and **Supplementary Fig. 9**). To this end, we rely on tools from the Bioconductor project for identifying differentially expressed transcripts, including edgeR[15] and DESeq[16]. Bioconductor requires the R software for statistical computing, which includes a command-line environment and programming language syntax that can present a barrier to new users or those lacking extensive bioinformatics training. To facilitate the use of Bioconductor tools for transcriptome studies, the Trinity software suite includes easy-to-use scripts that leverage the R software to identify differentially expressed transcripts; generate tab-delimited output files listing differentially expressed transcripts including fold change and statistical significance values; and generate visualizations such as MA plots (where M = log ratios and A = mean values), volcano plots, correlation plots and clustered heat maps in PDF format (see 'Protocol overview' and PROCEDURE).

Protein-coding region prediction and functional annotation of Trinity transcripts

Most transcripts assembled from eukaryotic RNA-seq data derived from polyadenylated RNA are expected to code for proteins. A sequence homology search, such as by BLASTX, against sequences from a well-annotated, phylogenetically related species is the most practical way to identify likely coding transcripts and to predict their functions. Unfortunately, such well-annotated 'relative' species are often not available for newly targeted transcriptomes. In such cases, using the latest nonredundant protein database (e.g., NCBI's 'nr', ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/nr.gz or Uniprot ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_trembl.fasta.gz) is an appropriate alternative.

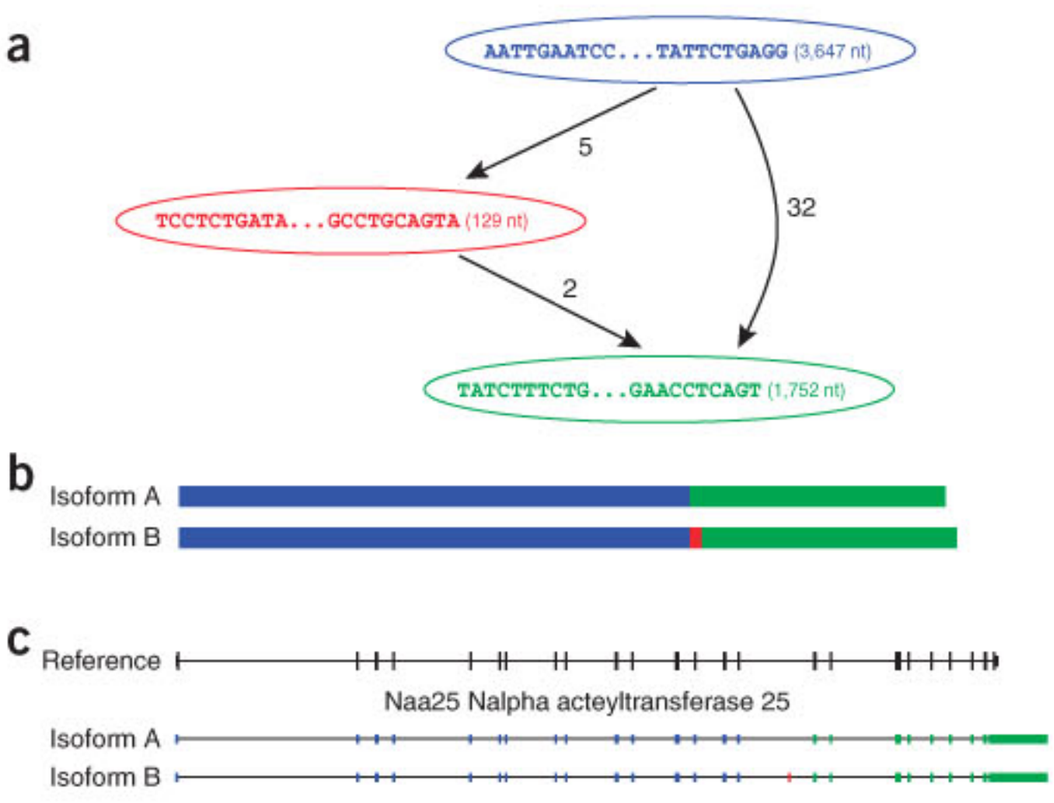

Figure 3 | Transcriptome and genome representations of alternatively spliced transcripts. (**a**–**c**) An example of the graphical representation generated by Trinity's Butterfly software (**a**) along with the corresponding reconstructed transcripts (**b**) and their exonic structure based on alignment to the mouse genome (**c**). Each node in **a** is associated with a sequence, and directed edges connect consecutive sequences from 5′ to 3′ in the same transcript. Bulges and bifurcations indicate sequence differences between alternative reconstructed transcripts, including alternatively spliced cassette exons; only a single bulge is shown in this transcript graph, yielding the red node. Edges are annotated by the number of RNA-seq fragments supporting the transcript from the 5′ sequence to the 3′ one. In this example, there are two supported paths: one from the blue to the green node (supported by 32 fragments) yielding 'isoform A' (**b**, top), and the other from the blue to the red to the green node, supported by at most five fragments, yielding 'isoform B' (**b**, bottom). The red node is a result of an alternatively skipped exon, as apparent in the gene structure (**c**, red bar, shown in 'isoform B'). Navigable transcript graphs are optionally generated by Butterfly, provided in 'dot' format, and can be visualized using graphviz (http://graphviz.org). These details are provided on the Trinity website (http://trinityrnaseq.sourceforge.net/advanced_trinity_guide.html).

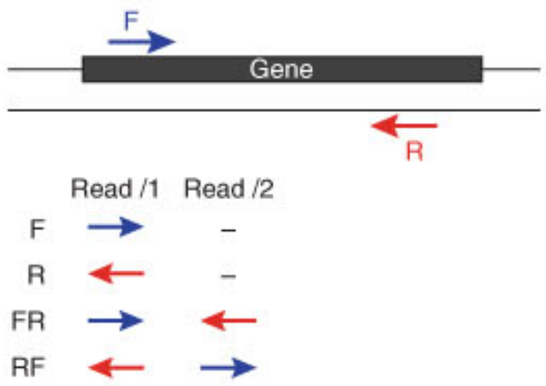

Figure 4 | Strand-specific library types. The left (/1) and right (/2) sequencing reads are depicted according to their orientations relative to the sense strand of a transcript sequence. The strand-specific library type (F, R, FR or RF) depends on the library construction protocol and is user-specified to Trinity via the *'--SS_lib_type'* parameter.

Newly targeted transcriptomes may also encode proteins that are insufficiently represented by detectable homologies to known proteins. Capturing those coding regions requires methods that predict coding regions based on metrics tied to sequence composition. One such utility is TransDecoder (**Supplementary Note**), which we developed and include with Trinity to assist in the identification of potential coding regions within reconstructed transcripts. When run on the Trinity-reconstructed transcripts, TransDecoder identifies candidate protein-coding regions on the basis of nucleotide composition, open reading frame length and (optional) Pfam domain content. Although running TransDecoder and functionally annotating coding regions are not covered as part of the present protocol, relevant documentation is provided in the Trinity website at http://trinityrnaseq.sourceforge.net/analysis/extract_proteins_from_trinity_transcripts.html and http://trinityrnaseq.sourceforge.net/annotation/Trinotate.html.

Dynamic graphical user interfaces, such as IGV[27] or GenomeView[28], are especially useful in studying transcript reconstructions. Although they were originally designed for genomes, these interfaces can be readily used to view alignments of reads to transcripts, putative coding regions, regions of protein sequence homology and short-read alignments with pair links (e.g., http://trinityrnaseq.sourceforge.net/analysis/read_alignment_visualization_QC.html).

Limitations of the Trinity approach to transcriptome analysis

Although Trinity is highly effective in reconstructing transcripts and alternatively spliced isoforms, in the absence of a reference genome it can be difficult, if not impossible, to fully understand the structural basis for the observed transcript variations, such as whether they are due to one or more skipped exons, alternative donor or acceptor spliced sites or retained introns. We are currently exploring experimental and computational strategies to better enable achieving such insights from RNA-seq data in the absence of reference genomes.

The RNA-seq reads and pairing information enables Trinity to resolve isoforms and paralogs[8], but its success depends on finding sequence variations that can be properly phased by individual reads or through pair links. Sequence variations that cannot be properly phased can result in erroneous chimeras between isoforms or paralogs that are impossible to discern from short-read data alone. Improvements in long-read technologies[29] should help address these challenges. Notably, although Trinity currently only officially supports Illumina RNA-seq data, efforts are underway to explore the use of transcript-sequencing reads generated from alternative technologies, including those from Pacific Biosciences[30] and Ion Torrent[31].

Finally, as in high-throughput genome sequencing, evidence for polymorphisms can be mined from the Illumina RNA-seq data mapped to Trinity assemblies (as in van Belleghem *et al.*[32]) and visualized within the display. However, researchers must be particularly cautious in evaluating polymorphisms in the context of RNA-seq and *de novo* transcriptome assembly data, as incorrect transcript assembly or isoform misalignment can be easily misinterpreted as evidence of polymorphism. In particular, when transcripts are very highly expressed, sequencing errors can yield substantially expressed 'variants'. Determining the best practices for calling SNPs in *de novo* transcriptome assemblies and examining allele-specific expression is an open area of research. Indeed, as bioinformatic software become easier to use, it is essential for the research community to develop and use best practices to ensure that controversial results are not the result of multiple sources of error[33].

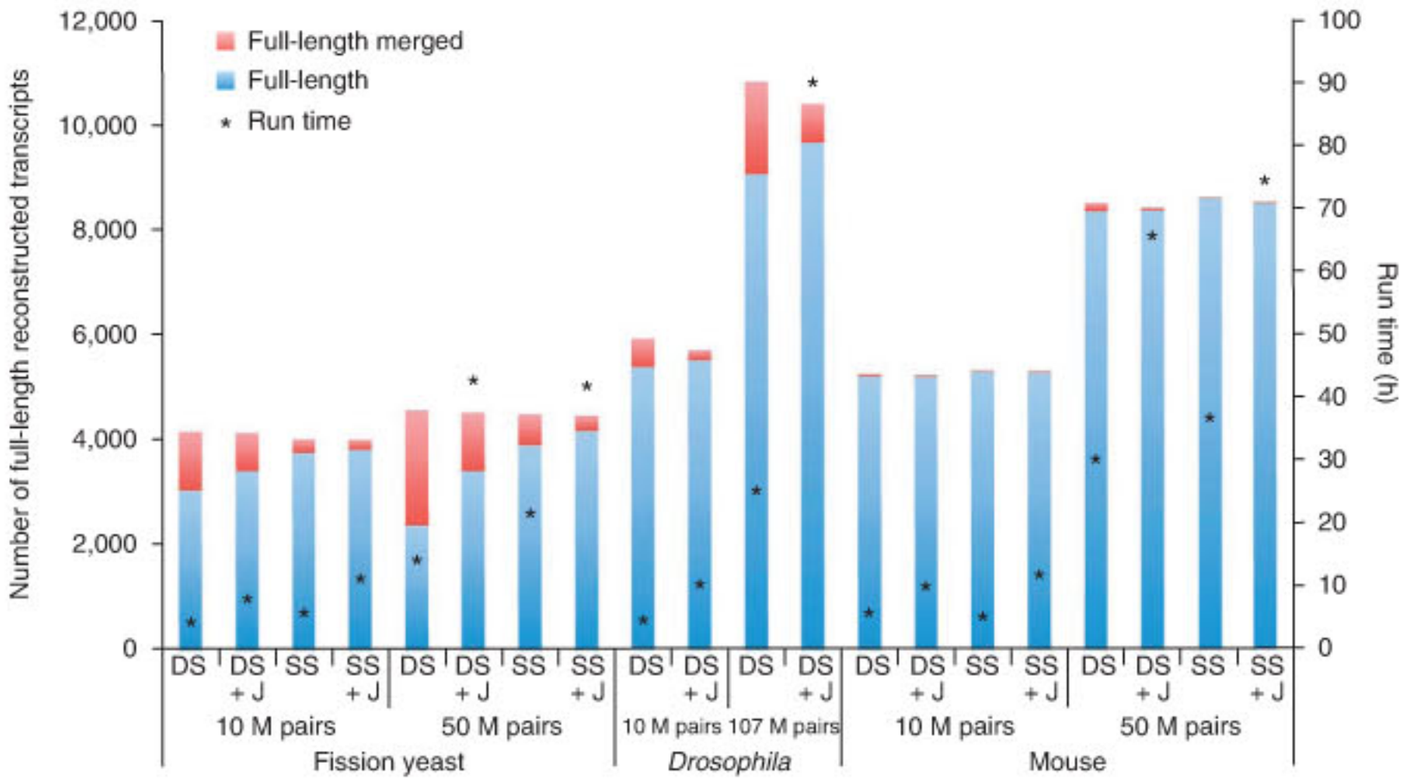

Figure 5 | Full-length transcript reconstruction by Trinity in different organisms, sequencing depths and parameters. The *y* axis to the left-hand side shows the number of fully reconstructed transcripts for Trinity assemblies of RNA-seq data derived from fission yeast (*S. pombe*[8,41]), *Drosophila melanogaster*[11] and mouse[8] with different combinations of parameters: DS, double-stranded mode; SS, strand-specific mode; +J, using the *'--jaccard_clip'* parameter to split falsely fused transcripts. Both SS and DS results are provided for *S. pombe* and mouse, but only DS results are provided for *Drosophila*, as its RNA-seq data were not strand specific. Blue shows full-length transcripts; red shows full-length merged transcripts (i.e., transcripts erroneously fused (multicistronic) with another (typically neighboring) transcript). The black asterisks (values shown in the *y* axis on the right-hand side of the graph) indicate the run times in each case with a contemporary high-memory (256–512 GB of RAM) server using a maximum of four threads (*'--CPU 4'*, see Step 6 of the PROCEDURE).

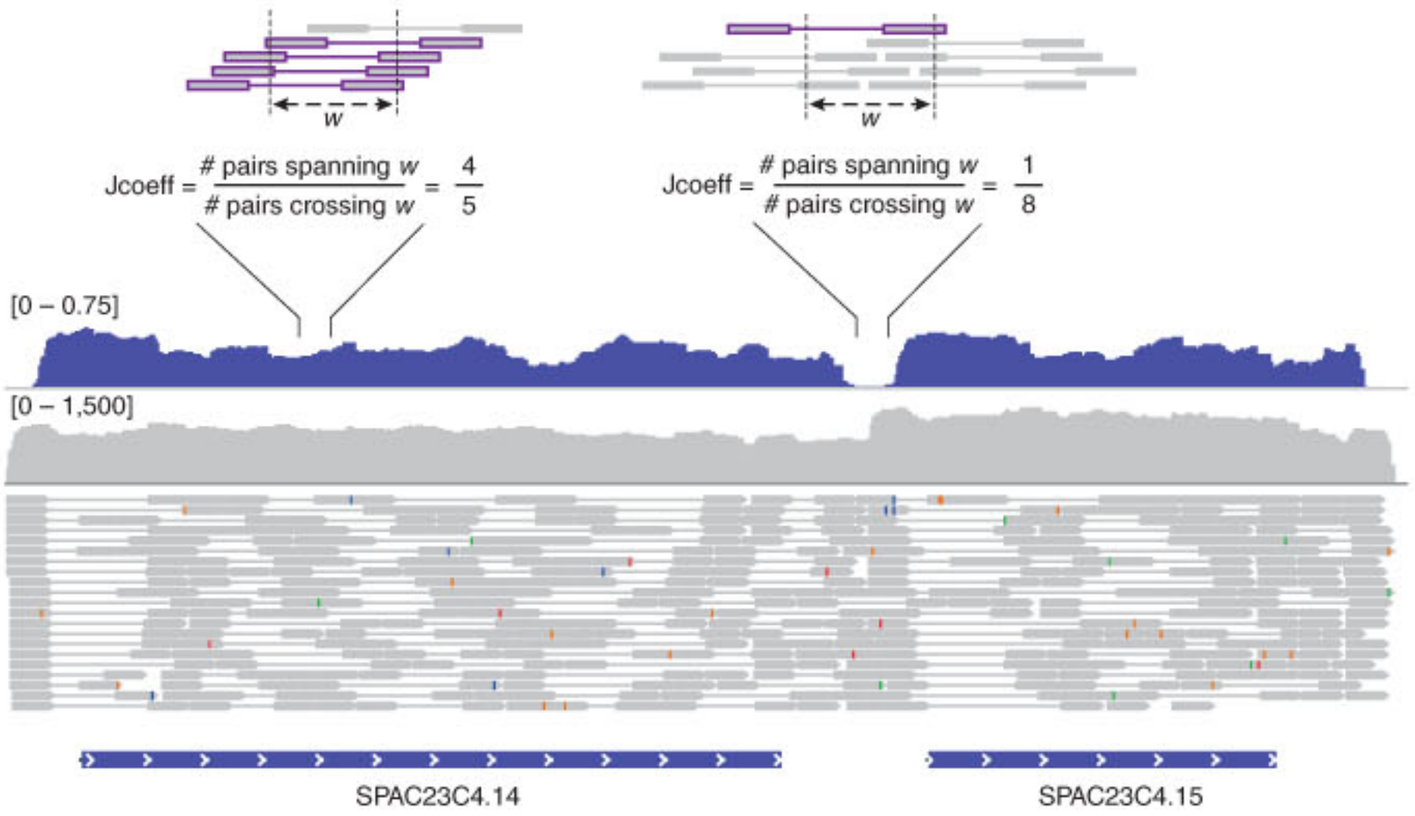

Figure 6 | Evaluating paired-read support via the Jaccard similarity coefficient. Read pair support is computed by first counting the number of RNA-seq fragments (bounds of paired reads) that span each of two outer points of a specified window length (default: 100 bases), and then computing the Jaccard similarity coefficient (intersection/union) comparing the fragments that overlap either point. An example is shown for a neighboring pair of *S. pombe* transcripts (SPAC23C4.14 and SPAC23C4.15, bottom) with substantial overlapping read coverage (gray track), resulting in a contiguous (fused) transcript assembled by Inchworm. However, the Jaccard similarity coefficient (blue track) calculated from the paired reads (gray dumbbells) clearly identifies the position of reduced pair support. Examples of strong (upper left) and weak (upper right) pair support are depicted at the top. When using the '*--jaccard_clip*' parameter, the Inchworm contig is dissected into two separate full-length transcripts, which are then further processed by Chrysalis and Butterfly as part of the Trinity pipeline.

Alternative analysis packages

Excellent tools are available and in widespread use for transcriptome studies in organisms for which a high-quality reference genome sequence is available, including those provided in the Tuxedo software suite (Bowtie, TopHat, Cufflinks, Cuffdiff and CummeRbund[4,21,23,34]) or Scripture[5]. Available *de novo* RNA-seq assembly software include, among others, Oases[7], SOAPdenovo-trans and TransABySS[6]. The eXpress software[22] implements a highly efficient algorithm for estimating transcript expression levels, and leverages the Bowtie2 software[34] for short-read alignments, providing an alternative to using RSEM for estimating transcript measurements. New methods for differential expression analysis based on RNA-seq data are also emerging[35–39]. As we continue to maintain and enhance the Trinity software and support related downstream analyses, we aim to explore the impact of new tools as they become available, and integrate those found to be most useful into future analysis pipelines, and we encourage users to explore alternative methods independently. In addition, we encourage users to explore the currently supported tools, including edgeR and RSEM, independently of using the Trinity-provided helper utilities, as they include additional capabilities that may not be fully accessible via the helper utility interface.

Future Trinity developments are planned to not only support genome-free *de novo* transcriptome assembly but also to be able to leverage reference genome sequences and transcript annotations, where available. In addition to providing effective methods to assist in genome annotation, such developments should expand researchers' ability to explore the transcriptional complexity of model organisms, particularly in those cases in which genomes are modified or rearranged, such as in cancer[40].

Protocol overview

The following procedure details how to run Trinity to assemble a transcriptome reference from RNA-seq from multiple samples; estimate expression levels for each transcript in each sample; and, finally, identify transcripts that are differentially expressed between the different samples.

This protocol provides a walk-through for some standard operations used to generate and analyze Trinity assemblies, including *de novo* RNA-seq assembly, abundance estimation and differential expression. For simplicity, libraries from biological replicates are not used in the protocol, but note that at least three biological

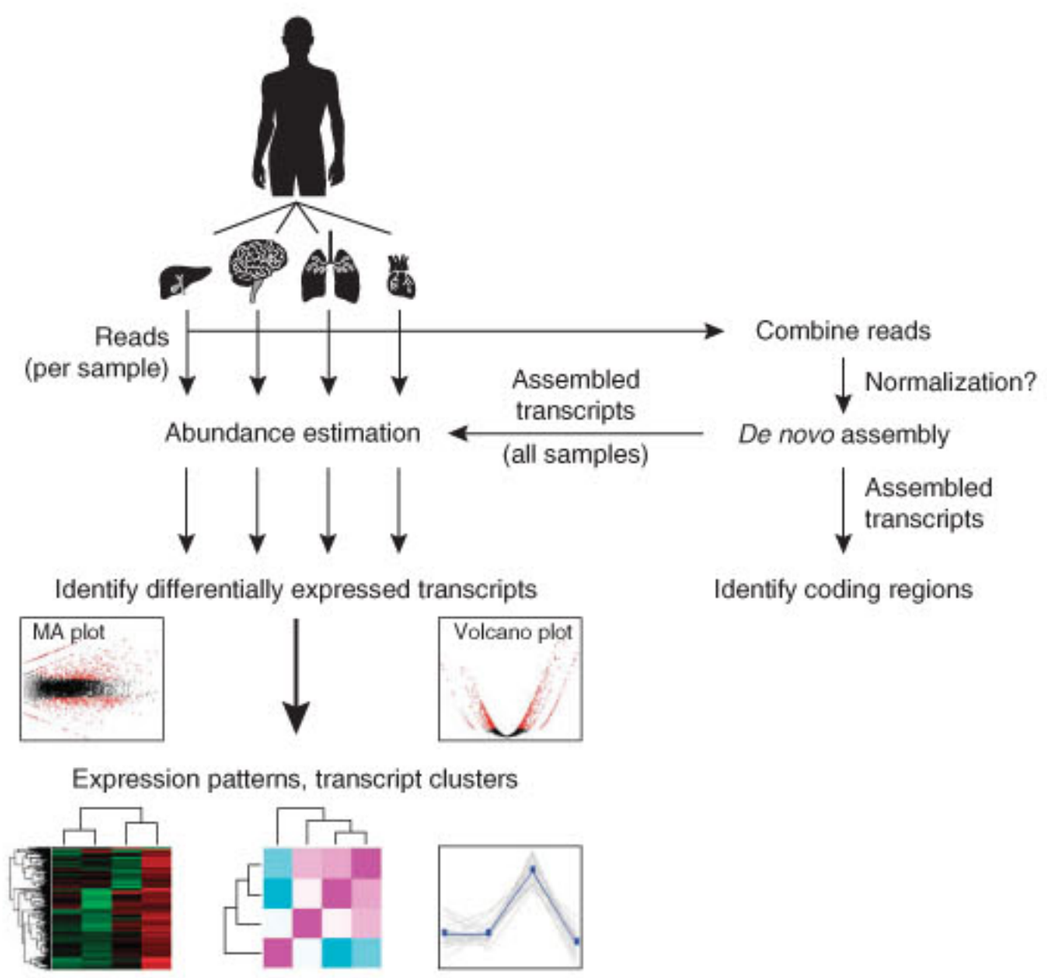

Figure 7 | *De novo* transcriptome assembly and analysis workflow. Reads from multiple samples (e.g., different tissues, top) are combined into a single data set. Reads may be normalized to reduce read counts while retaining read diversity and sample complexity. The combined read set is assembled by Trinity to generate a 'reference' *de novo* transcriptome assembly (right). Protein-coding regions can be extracted from the reference assembly using TransDecoder and further characterized according to likely functions based on sequence homology or domain content. Separately, sample-specific expression analysis is performed by aligning the original sample reads to the reference transcriptome assembly on a per sample basis, followed by abundance estimation using RSEM. Differentially expressed transcripts are identified by applying the Bioconductor software, such as edgeR, to a matrix containing the RSEM abundance estimates (number of RNA-seq fragments mapped to each transcript from each sample). Differentially expressed transcripts can then be further grouped according to their expression patterns.

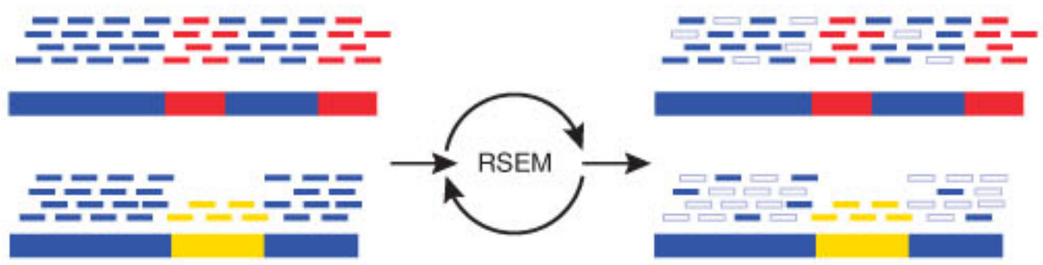

Figure 8 | Abundance estimation via expectation maximization by RSEM. An illustrative example of abundance estimation for two transcripts with shared (blue) and unique (red, yellow) sequences. To estimate transcript abundances, RNA-seq reads (short bars) are first aligned to the transcript sequences (long bars, bottom). Unique regions of isoforms will capture uniquely mapping RNA-seq reads (red and yellow short bars), and shared sequences between isoforms will capture multiply-mapping reads (blue short bars). An expectation maximization algorithm, implemented in the RSEM software, estimates the most likely relative abundances of the transcripts and then fractionally assigns reads to the isoforms based on these abundances. The assignments of reads to isoforms resulting from iterations of expectation maximization are illustrated as filled short bars (right), and eliminated assignments are shown as hollow bars. Note that assignments of multiply-mapped reads are in fact performed fractionally according to a maximum likelihood estimate. Thus, in this example, a higher fraction of each read is assigned to the more highly expressed top isoform than to the bottom isoform.

replicates per sample or condition are required in order to test for significance given observed biological and technical variation.

All the methods and tools for interrogating assemblies are described in the Trinity software website (http://trinityrnaseq.sf.net), and here we provide a selection of these operations that strikes a balance between the length of this article and showcasing the breadth of capabilities. This protocol is based on the use of version Trinityrnaseq_r2013-02-25 of the software, and readers should keep in mind that, as Trinity is a continually evolving research software, some parameters and filenames might change in future software releases. The most recent version of this procedure (tutorial) is maintained at http://trinityrnaseq.sf.net/trinity_rnaseq_tutorial.html. A typical use case for Trinity is not very different from the one described in this protocol, and we highly recommend users to perform the full procedure as detailed herein successfully before applying the protocol to their own data. Furthermore, before executing the steps described in the procedure, we encourage you to first read through the entire protocol.

MATERIALS

EQUIPMENT

▲ **CRITICAL** Ensure that each of the software tools mentioned in this section (except Trinity) is available within your Unix PATH setting. For example, if you have tools installed in a '/usr/local/tools' directory, you can update your PATH setting to include this directory in the search path by using the following command:

```
% export PATH=/usr/local/tools:$PATH
```

- RNA-seq data (see **Box 1** for details on input requirements and data preprocessing; please note that before applying the protocol to their own data, we highly recommend that users successfully perform the full procedure using the example data set provided)
- Hardware (64-bit computer running Linux; ~1 GB of RAM per ~1 million paired-end reads; **Box 2**)
- Trinity version trinityrnaseq_r2013-02-25 (http://trinityrnaseq.sourceforge.net)
- Bowtie version 0.12.9 (http://bowtie-bio.sourceforge.net; RSEM is currently not compatible with Bowtie 2)
- Samtools version 0.1.18 (http://sourceforge.net/projects/samtools/files/samtools/)
- R version 2.15 (http://www.r-project.org)
- Blast+ version 2.2.27 (ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/)

EQUIPMENT SETUP

Software setup Optionally, and only for simplicity, define the environmental variable TRINITY_HOME, replacing '/software/trinityrnaseq' below with the path to your Trinity software installation. Alternatively, write the full path where `$TRINITY_HOME` appears in the PROCEDURE.

```
% export TRINITY_HOME=/software/trinityrnaseq
```

After you install R, install the following R packages: Bioconductor (http://www.bioconductor.org), edgeR (http://www.bioconductor.org/packages/release/bioc/html/edgeR.html) and gplots, all as described below from within an R session:

```
% R
> source(" http://bioconductor.org/biocLite.R ")
> biocLite()
> biocLite (' edgeR ')
> biocLite(' ctc ')
> biocLite(' Biobase ')
> biocLite(' ape ')
> install.packages(' gplots ')
```

PROCEDURE

Collection of RNA-seq data ● TIMING ~10 min

1| Download strand-specific RNA-seq data from *Schizosaccharomyces pombe* grown in four conditions[41] (logarithmic growth, plateau phase, diauxic shift and heat shock, each with 1 million Illumina paired-end strand-specific RNA-seq data, for a total of 4 million paired-end reads) by visiting the URL reported below in a web browser, or directly from the command line using 'wget', by using the following command:

```
% wget \
  http://sourceforge.net/projects/trinityrnaseq/files/misc/TrinityNatureProtocol
  Tutorial.tgz/download
```

PROTOCOL

2| Name the file thus downloaded 'TrinityNatureProtocolTutorial.tgz', which should be 540 MB in size.

3| Unpack this file by using the following command:

```
tar -xvf TrinityNatureProtocolTutorial.tgz
```

This 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.
```

▲ **CRITICAL STEP** Please note that this protocol expects the raw data to be of high quality, free from adapters, barcodes and other contaminating sub-sequences.

De novo RNA-seq assembly using Trinity ● TIMING 60–90 min

4| Create a working folder and place the 'TrinityNatureProtocolTutorial' directory contents there (as per the Materials section).

5| 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 using the following commands:

```
% cat 1M_READS_sample/*.left.fq > reads.ALL.left.fq
% cat 1M_READS_sample/*.right.fq > reads.ALL.right.fq
```

6| Assemble the reads into transcripts using Trinity with the following commands:

```
% $TRINITY_HOME/Trinity.pl --seqType fq --JM 10G \
  --left reads.ALL.left.fq --right reads.ALL.right.fq \
  --SS_lib_type RF --CPU 6
```

Please note that the '--JM option' enables the user to control the amount of RAM used during Jellyfish *k*-mer counting: 10 GB in this case. The '--CPU' option controls the number of parallel processes. Feel free to change these parameters depending on your system. The Trinity-reconstructed transcripts will exist as FASTA-formatted sequences in the output file 'trinity_out_dir/Trinity.fasta'.

? TROUBLESHOOTING

7| Use the script '`$TRINITY_HOME/utilities/TrinityStats.pl`' to examine the statistic for the Trinity assemblies:

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

The 'TrinityStats.pl' reports the number of transcripts, components and the transcript contig N50 value on the basis of the 'Trinity.fasta' file. The contig N50 value, defined as the maximum length whereby at least 50% of the total assembled sequence resides in contigs of at least that length, is a commonly used metric for evaluating the contiguity of a genome assembly. Note that, unlike genome assemblies, maximizing N50 is not appropriate for transcriptomes; it is more appropriate to use an index

based on a reference data set (from the same or a closely related species) and to estimate the number of reference genes recovered and how many can be deemed to be full length[42,43]. The N50 value is, however, useful for confirming that the assembly succeeded (you will expect a value that is near the average transcript length of *S. pombe*; average = 1,397 bases).

Quality assessment (optional) ● **TIMING ~90 min**

▲ **CRITICAL** This section of the PROCEDURE is optional, but we highly recommend its implementation.

8| Examine the breadth of genetic composition and transcript contiguity by leveraging a reference data set. The annotated reference transcriptome of *S. pombe* is included as file 'S_pombe_refTrans.fasta'. Use megablast and our included analysis script to analyze its representation by the Trinity assembly as described below (Steps 9–13).

? TROUBLESHOOTING

9| Prepare the reference transcriptome FASTA file as a BLAST database:

```
% makeblastdb -in S_pombe_refTrans.fasta -dbtype nucl
```

10| Run megablast to align the known transcripts to the Trinity assembly:

```
% blastn -query trinity_out_dir/Trinity.fasta \
  -db S_pombe_refTrans.fasta \
  -out Trinity_vs_S_pombe_refTrans.blastn \
  -evalue 1e-20 -dust no -task megablast -num_threads 2 \
  -max_target_seqs 1 -outfmt 6
```

11| Once megablast has completed, run the script below to examine the length coverage of top database hits:

```
% $TRINITY_HOME/util/analyze_blastPlus_topHit_coverage.pl \
  Trinity_vs_S_pombe_genes.blastn \
  trinity_out_dir/Trinity.fasta \
  S_pombe_refTrans.fasta
```

12| Examine the number of input RNA-seq reads that are well represented by the transcriptome assembly. Trinity provides a script ('alignReads.pl') that executes Bowtie to align the left and right fragment reads separately to the Trinity contigs; it then groups the reads together into pairs while retaining those single-read alignments that are not found to be properly paired with their mates. Run 'alignReads.pl' as follows:

```
% $TRINITY_HOME/util/alignReads.pl --seqType fq \
  --left reads.ALL.left.fq --right reads.ALL.right.fq \
  --SS_lib_type RF --retain_intermediate_files \
  --aligner bowtie \
  --target trinity_out_dir/Trinity.fasta -- -p 4
```

13| When 'alignReads.pl' is run using strand-specific data, as indicated above with the '--SS_lib_type RF' parameter setting, it will separate the alignments that align to the sense strand ('+') from those that align to the antisense strand ('−'). All output files including coordinate-sorted and read-name–sorted SAM files should exist in a 'bowtie_out/' directory. Count the number of reads aligning (at least once) to the sense strand of transcripts by running the utility below on the sense-strand read name–sorted alignment file as shown:

```
% $TRINITY_HOME/util/SAM_nameSorted_to_uniq_count_stats.pl \
  bowtie_out/bowtie_out.nameSorted.sam.+.sam
```

Abundance estimation using RSEM ● **TIMING 40–60 min**

14| Obtain transcript abundance estimates by running RSEM separately for each sample, as shown below (Steps 15–18). The Perl script 'run_RSEM_align_n_estimate.pl' simply provides an interface to the RSEM software, translating the familiar Trinity

command-line parameters to their RSEM equivalents and then executing the RSEM software. Each relevant step below (Steps 15–18) generates files ('`${prefix}.isoforms.results`' and '`${prefix}.genes.results`') containing the abundance estimations for Trinity transcripts (**Table 1**) and components (**Table 2**), respectively. The `${prefix}` in the filename is set based on the '--prefix' setting in the commands below, which is unique to each sample. Please note that, with regard to the parameters reported in **Tables 1** and **2**, the gene length and effective length are defined as the IsoPct-weighted sum of transcript lengths and effective lengths. The gene expected counts, TPM and FPKM are defined as the sum of its transcripts' expected counts, TPM and FPKM.

? TROUBLESHOOTING

15| RSEM for log phase:

```
% $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl \
  --transcripts trinity_out_dir/Trinity.fasta \
  --left Sp.log.1M.left.fq \
  --right Sp.log.1M.right.fq \
  --seqType fq \
  --SS_lib_type RF \
  --prefix LOG
```

16| RSEM for diauxic shift:

```
% $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl \
  --transcripts trinity_out_dir/Trinity.fasta \
  --left Sp.ds.1M.left.fq \
  --right Sp.ds.1M.right.fq \
  --seqType fq \
  --SS_lib_type RF \
  --prefix DS
```

17| RSEM for heat shock:

```
% $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl \
  --transcripts trinity_out_dir/Trinity.fasta \
  --left Sp.hs.1M.left.fq \
  --right Sp.hs.1M.right.fq \
  --seqType fq \
  --SS_lib_type RF \
  --prefix HS
```

TABLE 1 | Example contents of RSEM's 'isoforms.results' file.

Transcript_id	Gene_id	Length	Effective length	Expected count	TPM	FPKM	IsoPct
comp56_c0_seq1	comp56_c0	3,739	3,443	637.65	16,664.43	7,008.23	11.26
comp56_c0_seq2	comp56_c0	3,697	3,401	4,966.34	131,393.38	55,257.53	88.74
comp62_c0_seq1	comp62_c0	7,194	6,898	4,551.13	59,364.09	24,965.59	95.54
comp62_c0_seq2	comp62_c0	7,076	6,778	208.87	2,771.95	1,165.74	4.46

'transcript_id' is the Trinity transcript identifier; 'gene_id' is the Trinity component to which the reconstructed transcript was derived; 'length' is the length of the reconstructed transcript; 'effective length' is the mean number of 5′ start positions from which an RNA-seq fragment could have been derived from this transcript, given the distribution of fragment lengths inferred by RSEM (the value is equal to transcript_length − mean_fragment_length + 1); 'expected count' is the number of expected RNA-seq fragments assigned to the transcript given maximum-likelihood transcript abundance estimates; 'TPM' is the number of transcripts per million; 'FPKM' is the number of RNA-seq fragments per kilobase of transcript effective length per million fragments mapped to all transcripts; and 'IsoPct' is the percentage of expression for a given transcript compared with all expression from that Trinity component.

TABLE 2 | Example contents of RSEM's 'genes.results' file.

Gene_id	Transcript_id(s)	Length	Effective length	Expected count	TPM	FPKM
comp56_c0	comp56_c0_seq1, comp56_c0_seq2	3,701.73	3,405.49	5,604	148,057.81	62,265.76
comp62_c0	comp62_c0_seq1, comp62_c0_seq2	7,188.74	6,892.5	4,760	62,136.04	26,131.33

18| RSEM for plateau phase:

```
% $TRINITY_HOME/util/RSEM_util/run_RSEM_align_n_estimate.pl \
  --transcripts trinity_out_dir/Trinity.fasta \
  --left Sp.plat.1M.left.fq \
  --right Sp.plat.1M.right.fq \
  --seqType fq \
  --SS_lib_type RF \
  --prefix PLAT
```

Differential expression analysis using edgeR ● **TIMING <5 min**

▲ **CRITICAL** Note that in this section the genes and transcripts can be examined separately using their corresponding RSEM abundance estimates. For brevity, we pursue here only the transcripts below.

19| You will see that each of the RSEM '*.isoforms.results' files has a number of columns, but we only need the one called 'expected_count'. Create a matrix containing the counts of RNA-seq fragments per feature in a simple tab-delimited text file using the expected fragment count data produced by RSEM.

```
% $TRINITY_HOME/util/RSEM_util/merge_RSEM_frag_counts_single_table.pl\
  LOG.isoforms.results DS.isoforms.results HS.isoforms.results \
  PLAT.isoforms.results > Sp_isoforms.counts.matrix
```

Please note that the first column of the resulting matrix is the name of the transcript. The second, third and so on are the raw counts for each of the corresponding samples. The first row contains the column headings including a label for each sample.

20| Use edgeR to identify differentially expressed transcripts for each pair of samples. The following script automates many of the tasks of running edgeR or DESeq; in this procedure, we only leverage edgeR. Use the matrix created in Step 19 as input.

```
% $TRINITY_HOME/Analysis/DifferentialExpression/run_DE_analysis.pl \
  --matrix Sp_isoforms.counts.matrix \
  --method edgeR \
  --output edgeR_dir
```

Please note that all the edgeR results from the pairwise comparisons now exist in the 'edgeR_dir/' output directory, and also include the following files of interest: *.edgeR.DE_results (differentially expressed transcripts identified, including fold change and statistical significance (**Table 3**)) and *.edgeR.DE_results.MA_n_Volcano.pdf (MA and volcano plots from pairwise comparisons (**Fig. 9**)).

21| To perform TMM normalization and to generate a matrix of expression values measured in FPKM, first extract the transcript length values from any one of RSEM's *.isoform.results files:

```
% cut -f1,3,4 DS.isoforms.results > Trinity.trans_lengths.txt
```

22| Now, perform the TMM normalization:

```
% $TRINITY_HOME/Analysis/
DifferentialExpression/run_TMM_
normalization_write_FPKM_matrix.pl \
   --matrix Sp_isoforms.counts.matrix \
   --lengths Trinity.trans_lengths.txt
```

This command will generate the following files: 'Sp_isoforms.counts.matrix.TMM_info.txt', containing the effective library size for each sample after TMM normalization; and 'Sp_isoforms.counts.matrix.TMM_normalized.FPKM', which contains normalized transcript expression values according to the transcript and sample, measured as FPKM**.** This matrix file will be used for clustering expression profiles for transcripts across samples and generating heat map visualizations, as described below.

TABLE 3 | Example contents of logarithmic versus plateau growth edgeR 'DE_results' file.

Transcript	logFC	logCPM	*P* value	FDR
comp5128_c0_seq1	10.3	11.1	2.13e-22	1.22e-18
comp5231_c0_seq1	10.0	10.9	1.10e-21	3.13e-18
comp5097_c0_seq1	8.7	11.3	5.72e-20	1.10e-16
comp1686_c0_seq1	9.2	10.4	1.01e-19	1.46e-16
comp1012_c0_seq1	8.3	11.5	2.8e-19	3.23e-16

FDR, false discovery rate.
logFC: $\log_2$(fold change) = $\log_2$(plateau_phase/logarithmic_growth).
logCPM: $\log_2$(counts per million).

23| To study expression patterns of transcripts or genes across samples, it is often useful to restrict analysis to those transcripts that are significantly differentially expressed in at least one pairwise sample comparison. Given a set of differentially expressed transcripts, extract their normalized expression values and perform hierarchical clustering to group together transcripts with similar expression patterns across samples, and to group together those samples that have similar expression profiles according to transcripts. For example, enter the 'edgeR_dir/' output directory and extract those transcripts that are at least fourfold differentially expressed with false discovery–corrected statistical significance of at most 0.001 by using the following commands:

```
% cd edgeR_dir/
% $TRINITY_HOME/Analysis/DifferentialExpression/analyze_diff_expr.pl \
   --matrix ./Sp_isoforms.counts.matrix.TMM_normalized.FPKM \
   -C 2 -P 0.001
```

Please note that the -C parameter takes the $\log_2$ (fold_change) cutoff, which in this case is $\log_2(4) = 2$. A number of files are generated, all with the prefix 'diffExpr.P0.001_C2' indicating the parameter choices: 'diffExpr.P0.001_C2.matrix' contains the subset of transcripts from the complete matrix 'matrix.TMM_normalized.FPKM' that were identified as differentially expressed, as defined by the specified thresholds. 'diffExpr.P0.001_C2.matrix.heatmap.pdf' contains a clustered heat map image showing the relationships among transcripts and samples (**Fig. 10a**) and a heat map of the pair-wise Spearman correlations between samples (**Fig. 10b**). 'diffExpr.P0.001_C2.matrix.R.all.RData' is a local storage of all the data generated during this analysis, which is used further down in the PROCEDURE (Step 25) with additional analysis tools.

24| Determine the number of such differentially expressed transcripts by counting the number of lines in the file by using the command:

```
% wc -l diffExpr.P0.001_C2.matrix
```

Subtract 1 so that you do not count the column header line as a transcript entry. Note that the 'analyze_diff_expr.pl' script will also directly report the number of differentially expressed transcripts identified at the given thresholds.

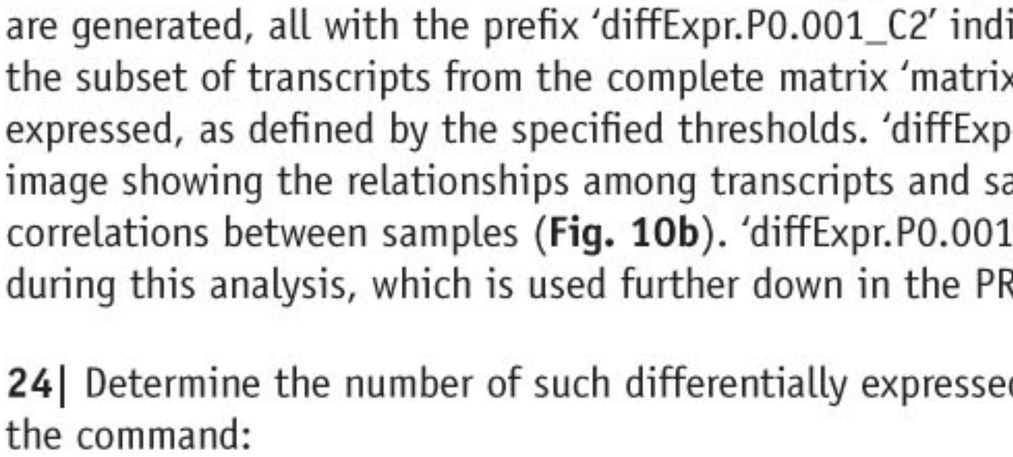

Figure 9 | Pairwise comparisons of transcript abundance. Two visualizations of the comparison of transcript expression profiles between the logarithmic growth and plateau growth samples from *S. pombe* to identify differentially expressed transcripts. (**a**) *MA* plot for differential expression analysis generated by EdgeR: for each gene, the $\log_2$(fold change) ($\log_2$(plateau_phase/logarithmic_growth)) between the two samples is plotted (*A*, *y* axis) against the gene's $\log_2$(average expression) in the two samples (*M*, *x* axis). (**b**) Volcano plot reporting false discovery rate ($-\log_{10}$FDR, *y* axis) as a function of $\log_2$ (fold change) between the samples (logFC, *x* axis). Transcripts that are identified as significantly differentially expressed at most 0.1% FDR are colored in red.

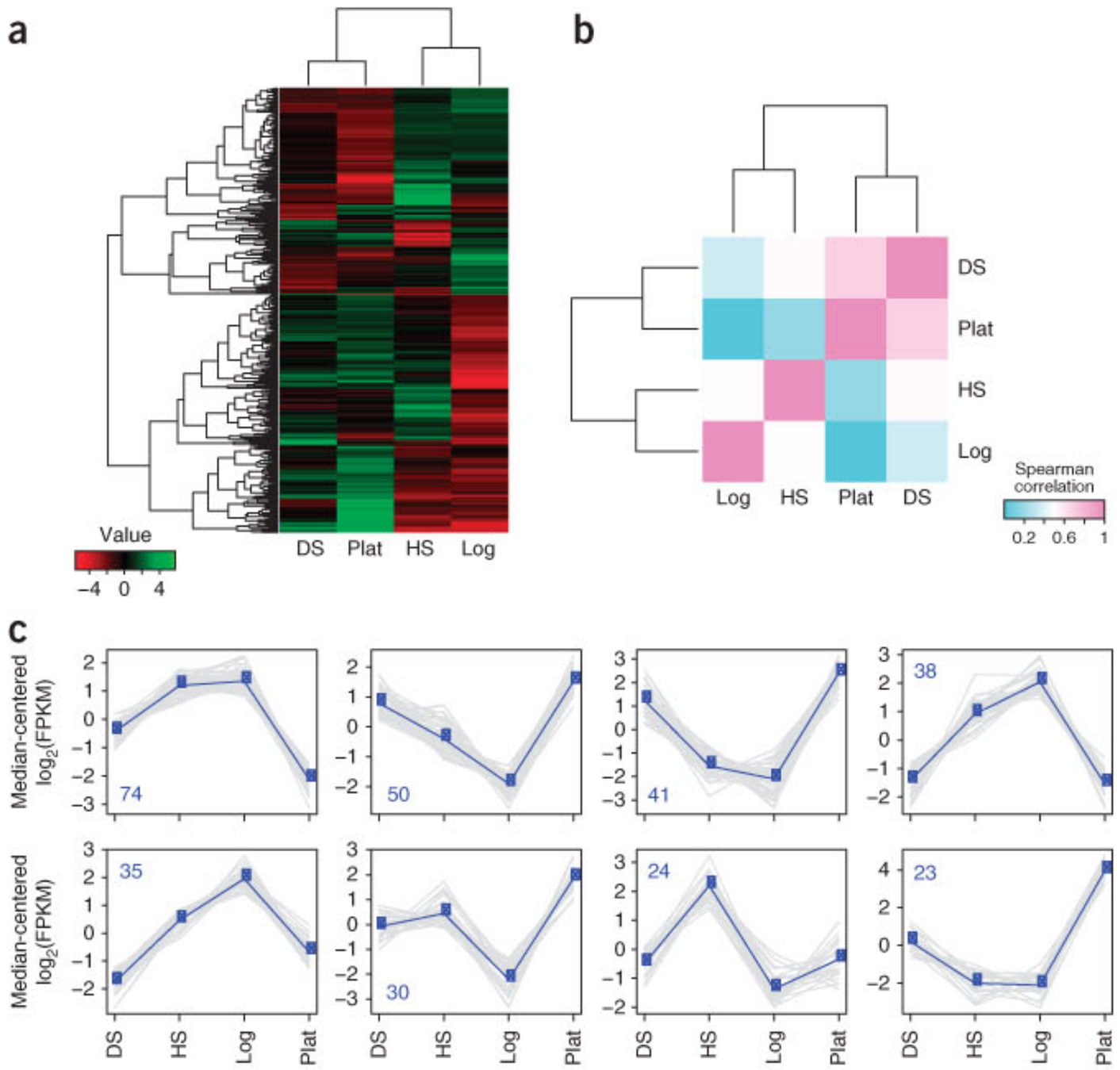

Figure 10 | Comparisons of transcriptional profiles across samples. (**a**) Hierarchical clustering of transcripts and samples. Shown is a heat map showing the relative expression levels of each transcript (rows) in each sample (column). Rows and columns are hierarchically clustered. Expression values (FPKM) are $\log_2$-transformed and then median-centered by transcript. (**b**) Heat map showing the hierarchically clustered Spearman correlation matrix resulting from comparing the transcript expression values (TMM-normalized FPKM) for each pair of samples. (**c**) Transcript clusters extracted from the hierarchical clustering with R. *X* axis: samples; *y* axis: median-centered $\log_2$(FPKM). Gray lines, individual transcripts; blue line, average expression values per cluster. Number of transcripts in each cluster is shown in the left corner of each plot. DS, diauxic shift; HS, heat shock; Log, mid-log growth; Plat, plateau growth.

25| Extract clusters of transcripts with common expression profiles from the earlier generated hierarchical clusters by running the script below, which uses R to cut the tree representing the hierarchically clustered transcripts based on specified criteria, such as to generate a specific number of clusters or by cutting the tree at a certain height. For example, run the following to partition transcripts by cutting the tree at 20% of its height:

```
% $TRINITY_HOME/Analysis/DifferentialExpression/define_clusters_by_cutting_tree.pl \
  --Ptree 20 -R diffExpr.P0.001_C2.matrix.R.all.Rdata
```

The above command generates a directory ('diffExpr.P0.001_C2.matrix.R.all.RData.clusters_fixed_P_20/') that contains 'subcluster_*_log2_medianCentered_fpkm.matrix'—each autodefined cluster of transcripts is provided along with expression values that are log2-transformed and median-centered—and 'my_cluster_plots.pdf'—contains a plot of the log2-transformed, median-centered expression values for each cluster (**Fig. 10c**). Note that, owing to the wide dynamic range in expression values of transcripts, during this step, the expression values were first $\log_2$-transformed before plotting data points. In addition, in order to examine common expression patterns that focus on the relative expression of transcripts across multiple samples, each transcript's expression value was subsequently centered by the median value. This operation was performed by subtracting each transcript's median $\log_2$(FPKM) value from its $\log_2$(FPKM) value in each sample. These resulting data are referred to as $\log_2$-transformed, median-centered expression values, as generated in this step.

26| (Optional) Run the script in Step 25 several times with different values of '--Ptree' in order to increase or decrease the number of clusters generated.

Automating the required sections of the PROCEDURE ● TIMING 2–3 h

27| (Optional) If you are interested in executing the sections of the PROCEDURE without manually typing in each command, run the script below, which executes the required sections of the PROCEDURE. These sections include concatenating all samples' reads into a single input data set, assembling the reads using Trinity, performing abundance estimation separately for each sample and running edgeR to identify differentially expressed transcripts. Run the automated procedure with the following command, including the -I (optional) parameter for an interactive experience, in which the system will pause and wait for a user response before proceeding to the next step.

```
% $TRINITY_HOME/util/run_Trinity_edgeR_pipeline.pl \
  --samples_file samples_n_reads_described.txt -I
```

? TROUBLESHOOTING

Troubleshooting advice can be found in **Table 4**.

TABLE 4 | Troubleshooting table.

Step	Problem	Possible reason	Solution
6	'bad_alloc' error	Insufficient computing resources resulting in a fatal out-of-memory error	Ensure that you have ~1 GB of RAM per ~1 million PE reads to be assembled. See **Box 2** for computing requirements and services available
	Large numbers (hundreds or more) of fusion transcripts	Not using strand-specific RNA-seq, or applying assembly to a transcriptome derived from a compact genome having (minimally) overlapping transcripts	If paired-end reads are being used, try running Trinity.pl with the '–jaccard_clip' parameter, which uses paired-end reads to separate minimally overlapping transcripts
	Retained introns are prevalent	Unprocessed RNA is captured and assembled, or contaminating genomic DNA contributes to the assembly	Setting Trinity.pl '–min_kmer_cov' to 2 or higher should reduce the number of retained introns, but will also reduce sensitivity for transcript reconstruction. Alternatively, transcripts with low expression (often enriched for retained introns) can be filtered from a given component postabundance estimation
8	Cannot find makeblastdb, blastn, or Bowtie	The additional required software tools were not installed or available via the Unix PATH setting	See EQUIPMENT section; be sure software tools are installed as required and that the software utilities are accessible via your PATH setting. Check with a systems administrator as necessary
14	Few or no transcripts identified as differentially expressed	Assuming transcripts are truly differentially expressed, insufficient sequencing depth caused the failure to detect differentially expressed transcripts	Adjust the sensitivity thresholds of 'analyze_diff_expr.pl', increasing the allowed FDR and lowering the fold-change requirements. Try running Bioconductor tools directly and examine the available options for data exploration. Increase your depth of sequencing to improve upon the detection of transcripts with low expression

● TIMING

Steps 1–3, collection of RNA-seq data: ~10 min
Steps 4–7, Trinity *de novo* transcriptome assembly of 4 million paired-end Illumina reads: 60–90 min
Steps 8–13, quality assessment (optional): ~90 min
Steps 14–18, using RSEM for abundance estimation: 40–60 min
Steps 19–25, differential expression analysis using EdgeR: <5 min
Step 26 (optional): variable
Step 27, automation of required sections: 2–3 h
The time taken for each of the commands executed for the required sections of the tutorial, as reported during the automated execution via 'run_Trinity_edgeR_pipeline.pl' described above, and as run on a high-performance server at the Broad Institute (hardware specifications included), is provided in the **Supplementary Note**.

TABLE 5 | Trinity assembly statistics for the assembly of 4 million paired-end *S. pombe* reads.

Assembly statistic	Value
Total Trinity transcripts	9,299
Total Trinity components	8,694
Contig N50	1,585

TABLE 6 | Distribution of BLASTN hit coverage of reference transcripts.

Length coverage bin (%)	Count of reference transcripts in bin	Cumulative count of reference transcripts at or above bin level
100	3,401	3,401
90	194	3,595
80	165	3,760
70	197	3,957
60	224	4,181
50	203	4,384
40	158	4,542
30	140	4,682
20	83	4,765
10	0	4,765
0	0	4,765

ANTICIPATED RESULTS

As Trinity's output is not absolutely deterministic, very slight variations in the output (number or length of transcripts) may result from Trinity being run at different times or on different hardware.

Trinity assembly statistics are provided in **Table 5**.

Reference transcript BLASTN mapping results from Step 11 are provided in **Table 6**. In all, 4,765 of the reference *S. pombe* transcripts have a BLAST hit with an *E*-value less than 1e-20, and 3,401 of the 5,163 total reference transcripts are considered to be of approximately 'full length', with the Trinity contigs aligning by greater than 90% of the matching reference transcript's length.

The counts of reads mapped to the Trinity assembly via alignReads.pl and using the Bowtie aligner, obtained in Step 13, are provided in **Table 7**.

The number of differentially expressed transcripts identified as having a significant false discovery rate (FDR) value of at most 0.001 and at least fourfold difference in expression values, ascertained from Step 24, is 659.

TABLE 7 | Counts of reads mapped to the Trinity assembly.

Read classification	Count of individual reads	Percentage of mapped reads
Proper pairing	8,102,100	93.12
Left only	307,933	3.54
Right only	284,203	3.27
Improper pairing	6,476	0.07

Note: Supplementary information is available in the online version of the paper.

ACKNOWLEDGMENTS We are grateful to D. Jaffe and S. Young for access to additional computing resources, to Z. Chen for help in R-scripting, to L. Gaffney for help with figure illustrations, to C. Titus Brown for essential discussions and inspiration related to digital normalization strategies, to G. Marcais and C. Kingsford for supporting the use of their Jellyfish software in Trinity and to B. Walenz for supporting our earlier use of Meryl. We are grateful to our users and their feedback, in particular J. Wortman and P. Bain for comments on earlier drafts of the manuscript. This project has been funded in part (B.J.H.) with Federal funds from the National Institute of Allergy and Infectious Diseases (NIAID), US National Institutes of Health (NIH), Department of Health and Human Services (DHHS), under contract no. HHSN272200900018C. Work was supported by Howard Hughes Medical Institute (HHMI), a NIH PIONEER award, a Center for Excellence in Genome Science grant no. 5P50HG006193-02 from the National Human Genome Research Institute (NHGRI) and the Klarman Cell Observatory at the Broad Institute (A.R.). A.P. was supported by the CSIRO Office of the Chief Executive (OCE). M.Y. was supported by the Clore Foundation. P.B. was supported by the National Science Foundation (NSF) grant no. OCI-1053575 for the Extreme Science and Engineering Discovery Environment (XSEDE) project. B.L. and C.D. were partially supported by NIH grant no.1R01HG005232-01A1. In addition, B.L. was partially funded by J. Thomson's MacArthur Professorship and by the Morgridge Institute for Research support for Computation and Informatics in Biology and Medicine. M.L. was supported by the Bundesministerium für Bildung und Forschung via the project 'NGSgoesHPC'. N.P. was funded by the Fund for Scientific Research, Flanders (Fonds Wetenschappelijk Onderzoek (FWO) Vlaanderen), Belgium. R.H. and R.D.L. were funded by the NSF under grant nos. ABI-1062432 and CNS-0521433 to Indiana University, and by Indiana METACyt Initiative, which is supported in part by Lilly Endowment, Inc. J.B. was supported through a CSIRO eResearch Accelerated Computing Project. Any opinions, findings and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of any of the funding bodies and institutions including the National Science Foundation, the National Center for Genome Analysis Support and Indiana University.

AUTHOR CONTRIBUTIONS B.J.H. is the current lead developer of Trinity and is additionally responsible for the development of the companion *in silico* normalization and TransDecoder utilities described herein. M.Y. contributed to Butterfly software enhancements, generating figures and to the manuscript text. B.L. and C.N.D. developed RSEM and are responsible for enhancements related to improved Trinity support. B.J.H. and A.P. wrote the initial draft of the manuscript. A.R. is the Principal Investigator. All authors contributed to Trinity development and/or writing of the final manuscript, and all authors approved the final text.

COMPETING FINANCIAL INTERESTS The authors declare no competing financial interests.

1. Wang, Z., Gerstein, M. & Snyder, M. RNA-seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.* **10**, 57–63 (2009).
2. Haas, B.J. & Zody, M.C. Advancing RNA-seq analysis. *Nat. Biotechnol.* **28**, 421–423 (2010).
3. Martin, J.A. & Wang, Z. Next-generation transcriptome assembly. *Nat. Rev. Genet.* **12**, 671–682 (2011).
4. Trapnell, C. *et al.* Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nat. Protoc.* **7**, 562–578 (2012).
5. Guttman, M. *et al. Ab initio* reconstruction of cell type–specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. *Nat. Biotechnol.* **28**, 503–510 (2010).
6. Robertson, G. *et al. De novo* assembly and analysis of RNA-seq data. *Nat. Methods* **7**, 909–912 (2010).
7. Schulz, M.H., Zerbino, D.R., Vingron, M. & Birney, E. Oases: robust *de novo* RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics* **28**, 1086–1092 (2012).
8. Grabherr, M.G. *et al.* Full-length transcriptome assembly from RNA-seq data without a reference genome. *Nat. Biotechnol.* **29**, 644–652 (2011).
9. Duan, J., Xia, C., Zhao, G., Jia, J. & Kong, X. Optimizing *de novo* common wheat transcriptome assembly using short-read RNA-seq data. *BMC Genomics* **13**, 392 (2012).
10. Xu, D.L. *et al. De novo* assembly and characterization of the root transcriptome of *Aegilops variabilis* during an interaction with the cereal cyst nematode. *BMC Genomics* **13**, 133 (2012).
11. Zhao, Q.Y. *et al.* Optimizing *de novo* transcriptome assembly from short-read RNA-seq data: a comparative study. *BMC Bioinformatics* **12** (suppl. 14), S2 (2011).
12. Henschel, R. *et al.* Trinity RNA-seq assembler performance optimization. XSEDE '12 Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: bridging from the eXtreme to the campus and beyond (Chicago, Illinois, USA, July 16–20, 2012) http://dx.doi.org/10.1145/2335755.2335842 (2012).
13. Marcais, G. & Kingsford, C. A fast, lock-free approach for efficient parallel counting of occurrences of *k*-mers. *Bioinformatics* **27**, 764–770 (2011).
14. Li, B. & Dewey, C.N. RSEM: accurate transcript quantification from RNA-seq data with or without a reference genome. *BMC Bioinformatics* **12**, 323 (2011).
15. 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).
16. Anders, S. & Huber, W. Differential expression analysis for sequence count data. *Genome Biol.* **11**, R106 (2010).

17. Bullard, J.H., Purdom, E., Hansen, K.D. & Dudoit, S. Evaluation of statistical methods for normalization and differential expression in mRNA-seq experiments. *BMC Bioinformatics* **11**, 94 (2010).
18. Fang, Z. & Cui, X. Design and validation issues in RNA-seq experiments. *Briefi. Bioinform.* **12**, 280–287 (2011).
19. Auer, P.L. & Doerge, R.W. Statistical design and analysis of RNA sequencing data. *Genetics* **185**, 405–416 (2010).
20. Mortazavi, A., Williams, B.A., McCue, K., Schaeffer, L. & Wold, B. Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nat. Methods* **5**, 621–628 (2008).
21. Trapnell, C. *et al.* Transcript assembly and quantification by RNA-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat. Biotechnol.* **28**, 511–515 (2010).
22. Roberts, A. & Pachter, L. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nat. Methods* **10**, 71–73 (2013).
23. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S.L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.* **10**, R25 (2009).
24. Robinson, M.D. & Oshlack, A. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol.* **11**, R25 (2010).
25. Dillies, M.A. *et al.* A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. *Brief. Bioinform.* http://dx.doi.org/10.1093/bib/bbs046 (17 September 2012).
26. Marioni, J.C., Mason, C.E., Mane, S.M., Stephens, M. & Gilad, Y. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Res.* **18**, 1509–1517 (2008).
27. Robinson, J.T. *et al.* Integrative genomics viewer. *Nat. Biotechnol.* **29**, 24–26 (2011).
28. Abeel, T., Van Parys, T., Saeys, Y., Galagan, J. & Van de Peer, Y. GenomeView: a next-generation genome browser. *Nucleic Acids Res.* **40**, e12 (2012).
29. Liu, L. *et al.* Comparison of next-generation sequencing systems. *J. Biomed. Biotechnol.* **2012**, 251364 (2012).
30. Eid, J. *et al.* Real-time DNA sequencing from single polymerase molecules. *Science* **323**, 133–138 (2009).
31. Rothberg, J.M. *et al.* An integrated semiconductor device enabling non-optical genome sequencing. *Nature* **475**, 348–352 (2011).
32. Van Belleghem, S.M., Roelofs, D., Van Houdt, J. & Hendrickx, F. *De novo* transcriptome assembly and SNP discovery in the wing polymorphic salt marsh beetle *Pogonus chalceus* (Coleoptera, Carabidae). *PLoS ONE* **7**, e42605 (2012).
33. Kleinman, C.L. & Majewski, J. Comment on "Widespread RNA and DNA sequence differences in the human transcriptome". *Science* **335**, 1302 (2012).
34. Langmead, B. & Salzberg, S.L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357–359 (2012).
35. Pounds, S.B., Gao, C.L. & Zhang, H. Empirical Bayesian selection of hypothesis testing procedures for analysis of sequence count expression data. *Stat. Appl. Genet. Mol. Biol.* http://dx.doi.org/10.1515/1544-6115.1773 (2012).
36. Tarazona, S., Garcia-Alcalde, F., Dopazo, J., Ferrer, A. & Conesa, A. Differential expression in RNA-seq: a matter of depth. *Genome Res.* **21**, 2213–2223 (2011).
37. Cumbie, J.S. *et al.* GENE-counter: a computational pipeline for the analysis of RNA-seq data for gene expression differences. *PLoS ONE* **6**, e25279 (2011).
38. Hardcastle, T.J. & Kelly, K.A. baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC Bioinformatics* **11**, 422 (2010).
39. Leng, N. *et al.* An empirical Bayes hierarchical model for inference in RNA-seq experiments. *Bioinformatics* **29**, 1035–1043 (2012).
40. Tuna, M. & Amos, C.I. Genomic sequencing in cancer. *Cancer Lett.* http://dx.doi.org/doi:10.1016/j.canlet.2012.11.004 (2012).
41. Rhind, N. *et al.* Comparative functional genomics of the fission yeasts. *Science* **332**, 930–936 (2011).
42. Kumar, S. & Blaxter, M.L. Comparing *de novo* assemblers for 454 transcriptome data. *BMC Genomics* **11**, 571 (2010).
43. Papanicolaou, A., Stierli, R., Ffrench-Constant, R.H. & Heckel, D.G. Next generation transcriptomes for next generation genomes using est2assembly. *BMC Bioinformatics* **10**, 447 (2009).
44. Lohse, M. *et al.* RobiNA: a user-friendly, integrated software solution for RNA-seq–based transcriptomics. *Nucleic Acids Res.* **40**, W622–W627 (2012).
45. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal* **17** http://journal.embnet.org/index.php/embnetjournal/article/view/200/479 (2011).
46. Haas, B.J., Chin, M., Nusbaum, C., Birren, B.W. & Livny, J. How deep is deep enough for RNA-seq profiling of bacterial transcriptomes? *BMC Genomics* **13**, 734 (2012).
47. 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).
48. Borodina, T., Adjaye, J. & Sultan, M. A strand-specific library preparation protocol for RNA sequencing. *Methods Enzymol.* **500**, 79–98 (2011).
49. Parkhomchuk, D. *et al.* Transcriptome analysis by strand-specific sequencing of complementary DNA. *Nucleic Acids Res.* **37**, e123 (2009).
50. Sung, W.K. *et al.* Genome-wide survey of recurrent HBV integration in hepatocellular carcinoma. *Nat. Genet.* **44**, 765–769 (2012).