PLOS ONE

RESEARCH ARTICLE

RefShannon: A genome-guided transcriptome assembler using sparse flow decomposition

Shunfu Mao[1], **Lior Pachter**[2], **David Tse**[3], **Sreeram Kannan**[1]*

1 Department of Electrical and Computer Engineering, University of Washington, Seattle, WA, United States of America, **2** Division of Biology and Biological Engineering, Caltech, Pasadena, CA, United States of America, **3** Department of Electrical Engineering, Stanford University, Stanford, CA, United States of America

* ksreeram@ece.uw.edu

OPEN ACCESS

Citation: Mao S, Pachter L, Tse D, Kannan S (2020) RefShannon: A genome-guided transcriptome assembler using sparse flow decomposition. PLoS ONE 15(6): e0232946. https://doi.org/10.1371/journal.pone.0232946

Editor: Zhong-Hua Chen, University of Western Sydney, AUSTRALIA

Received: October 18, 2019

Accepted: April 24, 2020

Published: June 2, 2020

Data Availability Statement: All relevant data are within the manuscript and its Supporting Information files.

Funding: This project is funded by NIH award 1R01HG008164, NSF CCF-1651236, and NSF CIF-1703403.

Competing interests: The authors have declared that no competing interests exist.

Abstract

High throughput sequencing of RNA (RNA-Seq) has become a staple in modern molecular biology, with applications not only in quantifying gene expression but also in isoform-level analysis of the RNA transcripts. To enable such an isoform-level analysis, a transcriptome assembly algorithm is utilized to stitch together the observed short reads into the corresponding transcripts. This task is complicated due to the complexity of alternative splicing - a mechanism by which the same gene may generate multiple distinct RNA transcripts. We develop a novel genome-guided transcriptome assembler, RefShannon, that *exploits the varying abundances* of the different transcripts, in enabling an accurate reconstruction of the transcripts. Our evaluation shows RefShannon is able to improve sensitivity effectively (up to 22%) at a given specificity in comparison with other state-of-the-art assemblers. RefShannon is written in Python and is available from Github (https://github.com/shunfumao/RefShannon).

Introduction

In many higher organisms including mammals, the genomic DNA is comprised of thousands of genes, along with other sequences that regulate when, where and how the genes are produced. These genes have distinct regions called exons and introns [1, 2]. By combining its exons in various ways—a procedure called alternative splicing [3–5], a gene, especially in eukaryotes, can produce multiple different messenger RNAs (mRNA [6], or RNA transcript in this document) that can be converted into different protein products in cells.

Transcriptome is the set of RNA transcripts. With RNA sequencing technology [7], we can now obtain millions of short RNA fragments (RNA-seq reads) from the transcriptome. The transcriptome assembly [8] problem is to obtain a complete and accurate recovery of transcriptome based on observed RNA-seq reads. This helps us find new RNA transcripts as well as their expression levels (or abundance) in order to better understand proteins and cells.

Transcriptome assembly is not an easy task due to several factors. Because of alternative splicing, it is possible that different set of transcripts can yield the same observation of RNA-seq reads. Adding to this complexity is the fact that distinct transcripts are expressed at

different expression levels [9]. Thus transcripts which have low expression levels are harder to reconstruct; also, transcripts which are part of complex isoform [4] families are difficult to assemble correctly.

There are two flavors of the transcriptome assembly problem [8]: de novo assembly and genome-guided (or reference-based) assembly. For de novo assembly, there is no knowledge other than that of the observed reads. This is common in non-model organisms or when an approach unbiased by prior knowledge is needed (e.g. cancer transcriptome). For genome-guided assembly, in addition to the observed reads there is also knowledge of the genome of the organism. This is common in scenarios where a model organism [10] is sequenced.

De novo assembly is typically more challenging and less accurate than genome-guided assembly, since the latter utilizes additional side information. While algorithms and software packages are available for both the de novo (TransAByss [11], Trinity [12], OASES [13], SOAPdenovo-Trans [14] etc) and genome-guided assembly problems (Scripture [15], Cufflinks [16], StringTie [17], TransComb [18] and CLASS2 [19], Ryuto [20], Strawberry [21], Trinity (reference-guided mode) [12] etc), much remains to be done [22, 23]. Recently Kannan et al [24] developed an assembler called Shannon assembler that utilized principles from information theory to solve the de novo transcriptome assembly problem, and demonstrated benefits over state-of-the-art assemblers. Here we develop a genome-guided assembler called RefShannon by exploiting the framework from Shannon.

To begin with we note that, while genome-guided algorithms have an advantage in general, Shannon is able to recover even more transcripts than the leading genome-guided assembler StringTie when the coverage of the transcripts is high (S1 File). We attribute this to the careful utilization of transcript abundances while performing assembly in Shannon. However, as expected, for transcripts of low abundance, Shannon is inferior because the k-mer graph utilized by Shannon needs a higher coverage in order to stay connected. Therefore, it should be possible to design a genome-guided assembler, that combines the superior reconstruction method of Shannon along with the aid of the genome side-information in order to deliver optimal performance. This is the main motivation for this work—to build a superior genome-guided assembler.

Results

In this section, we will describe the overall work flow of RefShannon, and highlight its main ideas. To demonstrate its superior performance, we have compared RefShannon to two widely used assemblers StringTie [17] and Cufflinks [15], which are recommended in previous benchmark work [23]. We have also compared RefShannon to guided Trinity [12] and recently published Ryuto [20], as they show relatively good performance among various assemblers in our initial analysis using smaller datasets (S3 File). Our performance evaluation metrics include ROC (including sensitivity and false positive) for simulated datasets and sensitivity for real datasets. Lastly, we also discuss its computational complexity.

Overall workflow

To do genome-guided transcriptome assembly based on sampled RNA-Seq reads, RefShannon takes a graph preparation step and a graph traversal step, as other assembly methods [16–21] usually do.

In particular as Fig 1 illustrates, RNA-Seq reads sampled from transcriptome will be aligned onto a reference genome using external tools such as STAR [25], Tophat2 [26], Hisat2 [27], GMAP [28], minimap2 [29] and so on. These tools are able to capture splice events (e.g. a read crossing two distinct exons of an RNA transcript can be split aligned onto the genome), which

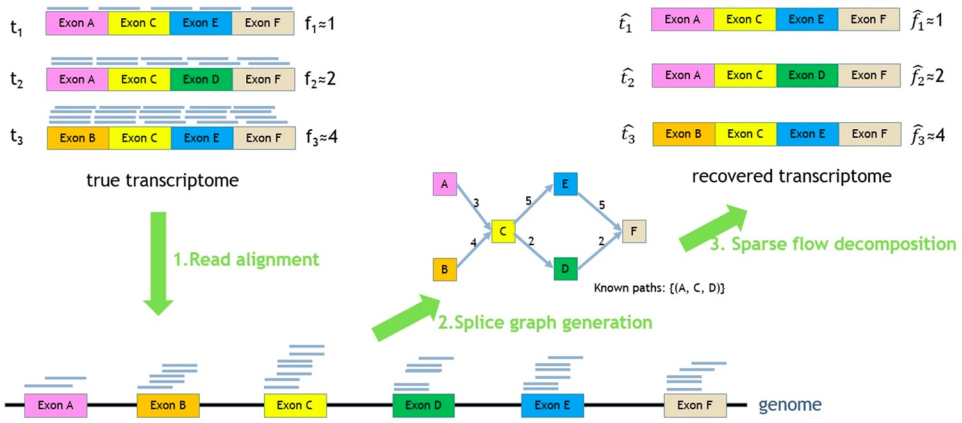

Fig 1. Overall flow of RefShannon. In this example, the true transcriptome contain 3 RNA transcripts t_1, t_2, t_3 with expression levels $f_1 = 1, f_2 = 2, f_3 = 4$. RNA reads are sampled from the transcriptome first, and are aligned onto a reference genome using external aligners. RefShannon will take the read alignment as input, generate splice graphs, and apply a novel sparse flow decomposition algorithm to recover the transcriptome as $\hat{t}_i$ $i \in \{1, 2, 3\}$. The corresponding $\hat{f}_i$ estimates related abundance.

https://doi.org/10.1371/journal.pone.0232946.g001

enable us to infer exon regions from read alignments and consequently do not need existing gene annotations for transcriptome assembly.

Based on read alignments, RefShannon produces splice graphs, where each node represents a unique exonic region that is supported by aligned reads and each edge between two nodes implies there exist reads going through those nodes. Additional known paths are also collected, indicating that the nodes along the path belong to some transcript. This helps resolve flow decomposition ambiguity, as will be discussed later.

Based on splice graphs, RefShannon applies a sparse flow decomposition algorithm, originally proposed in [24], to reconstruct the minimum number of flow paths (as assembled transcriptome) that satisfy node and edge constraints.

Main ideas

We would like to highlight several main ideas (as summarized in Fig 2) that distinguish RefShannon from the existing methods (such as Cufflinks [16], StringTie [17], Ryuto [20] and guided Trinity [12] which we'll mainly compare to) and bring its superior performance.

To begin with, RefShannon takes a graph preparation step and graph traversal step as most existing methods do. The constructed graph is consistent with the reference genome, similar to Cufflinks, Stringtie and Ryuto. Trinity in genome-guided mode also utilizes the reference genome, but mainly to group together the reads within the same region. Essentially it still applies de novo assembly, but to smaller regions. Consequently, guided Trinity is computationally much more complicated (S7 File).

One of the main ideas of RefShannon is to utilize the varying abundance information while traversing the splice graph (Fig 2A). This information is essential for a correct decomposition of flow paths. Cufflinks [16] does not exploit the abundance information during assembly; Instead, it relies on overlap graphs, which require a strict partial order between read alignments. Consequently Cufflinks may throw away reads (especially pair end reads) of uncertain compatibility, while those read alignments can contain useful information to construct a more accurate graph.

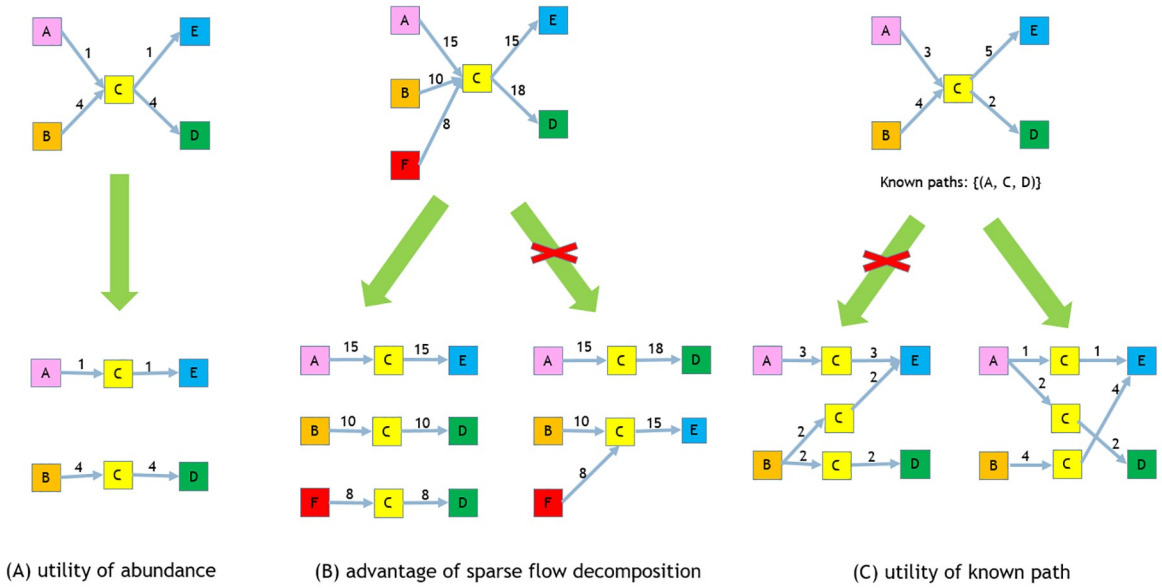

Fig 2. Main ideas of RefShannon. In these conceptual examples, each graph means a splice graph where nodes represent exonic regions and edges indicate there are reads aligned across the nodes. The edge weights are abundance, calculated by the number of supporting reads. (A) RefShannon explores the utility of varying abundance, which is essential for a correct decomposition of paths. For example, given that AC, CE, BC, CD have weights 1, 1, 4, 4 respectively, we're confident to decompose the graph into two paths as ACE and BCD. (B) RefShannon adopts a sparse flow algorithm that tries to find the minimum number of paths that explains edge weight constraints (e.g. the decomposition of graph into ACE with weight 15, BCD with weight 10 and FCD with weight 8 explain the splice graph well). If we follow a greedy approach (as used by StringTie) that iteratively finds and remove the heaviest path, we will get an inaccurate transcript ACD first. (C) RefShannon extracts known path information from read alignments to resolve decomposition ambiguity. For example, when node C is decomposed, a flow ambiguity (i.e. non unique decomposition) happens: both ACE, BCE, BCD and ACE, ACD, BCE explain node C's edge weight constraints. Suppose ACD is a known path, then the first decomposition of ACE, BCE, BCD can be excluded.

https://doi.org/10.1371/journal.pone.0232946.g002

Another idea RefShannon has adopted is the sparse flow decomposition algorithm that was initially proposed in the Shannon assembler [24], which applies linear programming to efficiently decompose for the minimum number of paths (flows) at each node restricted by the node's in-edge and out-edge weights. This approach has been proved in [24] to work toward optimal transcriptome assembly. The similar goal of parsimonious assembly is pursued by Cufflinks [16], which as mentioned earlier is based on overlap graph and does not exploit abundance information. StringTie [17] also explores the abundance information, but it follows a sub-optimal greedy approach by iteratively extracting the heaviest path as transcript. As illustrated in Fig 2B, this will lead to incorrect assembly.

In addition RefShannon has utilized pair end reads to supplement additional edge and path information so that an originally disconnected transcript could be found. A similar idea was also used in Scripture [15] and Ryuto [20]. However Scripture exhaustively enumerates all possible transcript candidates first and filter them later based on certain significance criteria. Ryuto has also adopted these information mainly during the graph construction stage. Differently, RefShannon has also applied these additional edge and path information in the graph decomposition stage so that certain assembly ambiguity can be resolved (Fig 2C). StringTie [17] on the other hand, does not explore these information directly; instead it relies on external software MaSuRCA [30] to generate super reads to fully utilize pair end reads.

Datasets

Our performance evaluation is based on both simulated datasets and real datasets, as summarized in Table 1.

The real datasets include 132.05M Illumina single end reads (50-bp) sampled from human embryonic stem cells (HESC) (GSE51861 [31]) with 13274 reference transcripts, 115.36M Illuminar pair end reads (101-bp) sampled from Lymphoblastoid cells (LC) (SRP036136 [32]) with 207266 reference transcripts. We also use 183.53M Illuminar pair end reads (100-bp) sampled from HEK293T (Kidney) cells (SRX541227) previously produced and studied in StringTie [17]. The HESC reference transcripts are assembled by hybrid assembler IDP from the 135M short reads together with 7.8M long PacBio reads [31]. The LC reference transcripts are based on GENCODE annotations augmented by utilizing a combination of short reads with long PacBio reads line (700K CCS reads) [32]. We also use LC reference transcripts for Kidney dataset.

The simulated datasets are generated based on the real ones. Specifically, we choose LC reference transcripts as the ground truth transcriptome for all simulated datasets. We then use RSEM [33] to learn parameters from each real dataset based on the alignments of real reads onto the reference transcripts. Based on the gound truth and learned parameters, we finally generate the relevant simulated reads.

Experiment procedure

The experiment procedure is as follows. For each dataset, we first use STAR aligner [25] to align reads onto reference genome (human genome hg19, downloaded from http://hgdownload.cse.ucsc.edu/goldenpath/hg19/bigZips/) that contains multiple chromosomes. We choose STAR aligner since it offers better performance for most of the assemblers in our evaluation than using alternative aligners such as Tophat2 [26] and Hisat2 [27] (S6 File); in addition different aligners do not affect the comparison conclusions (S3 File). We then apply assemblers onto read alignments to reconstruct transcripts for all chromosomes. The assemblers which we have selected to compare RefShannon to include Cufflinks (v2.2.1), StringTie (v1.3.4d), Ryuto (v1.3m) and Trinity (v2.9.1) as they show relatively good performance in our initial analysis of various assemblers using smaller datasets (S3 File). Finally we evaluate the reconstructed transcripts, and compare them with relevant reference transcripts to see the performance of ROC (including sensitivity and false positive) for simulated datasets and sensitivity for real datasets.

ROC of simulated datasets

For simulated datasets (HESC-Sim, LC-Sim, Kidney-Sim in Table 1), since we know the ground truth reference transcripts, we check the performance of receiver operating characteristic curves (ROC), which includes sensitivity as well as false positive.

Table 1. Data statistics. There're three real datasets (HESC—human embryonic stem cells, LC—Lymphoblastoid cells, Kidney) and three simulated datasets (HESC-Sim, LC-Sim, Kidney-Sim) used for evaluation. PE stands for pair-end reads and SE for single-end reads. Oracle Set contains fully covered transcripts from the available reference transcripts.

	HESC	LC	Kidney	HESC-Sim	LC-Sim	Kidney-Sim
PE or SE	SE	PE	PE	SE	PE	PE
Read Length	50	101	100	50	101	100
Num of Reads (SE) or Read Pairs (PE)	132.05M	115.36M	183.53M	150M	150M	150M
Num of Reference Transcripts	13274	207266	207266	207266	207266	207266
Num of Reference Transcripts (Oracle Set)	2694	32394	15605	-	-	-

https://doi.org/10.1371/journal.pone.0232946.t001

The metric of sensitivity describes how many reference transcripts have been correctly reconstructed. To evaluate sensitivity, we first use blat (https://genome.ucsc.edu/goldenpath/help/blatSpec.html) to create a mapping between reconstructed transcripts T_{rec} and reference transcripts T_{ref}. We associate each reference transcript $t \in T_{ref}$ with a reconstructed transcript $r \in T_{rec}$ that mostly matches t. We then consider each $t \in T_{ref}$ is correctly recovered if it is over 90% matched with its associated $r \in T_{rec}$.

The metric of false positive describes how many transcripts are falsely reconstructed and do not belong to the true reference transcripts. Based on the blat result, we consider a reconstructed transcript $r \in T_{rec}$ to be a false positive if it is below 90% matched with any reference transcript $t \in T_{ref}$. Note if $r \in T_{rec}$ is contained in a reference transcript $t \in T_{ref}$, it is not considered to be a false positive.

In addition to the threshold of 90%, we have also tried other thresholds, and it does not affect our comparison conclusions (S5 File).

To have a comprehensive understanding of the ROC performance, we run StringTie and Cufflinks in both default mode and max sensitivity mode; we run Ryuto and guided Trinity in their default modes after tuning several parameters which don't affect sensitivity much (S4 File). We also tune RefShannon so that it is able to adaptively adjust its splice graph generation as well as sparse flow decomposition output in order to trade off its sensitivity and false positive performance (S2 File).

Fig 3 illustrates the ROC performance. Overall RefShannon shows higher sensitivity at a given false positive ratio than other assemblers. For example in simulated LC dataset, the max sensitivity point of RefShannon (the top right blue point) has higher sensitivity and lower false positive than StringTie in max sensitivity setting and the min false positive point of RefShannon (the lower left blue point) has higher sensitivity and lower false positive than

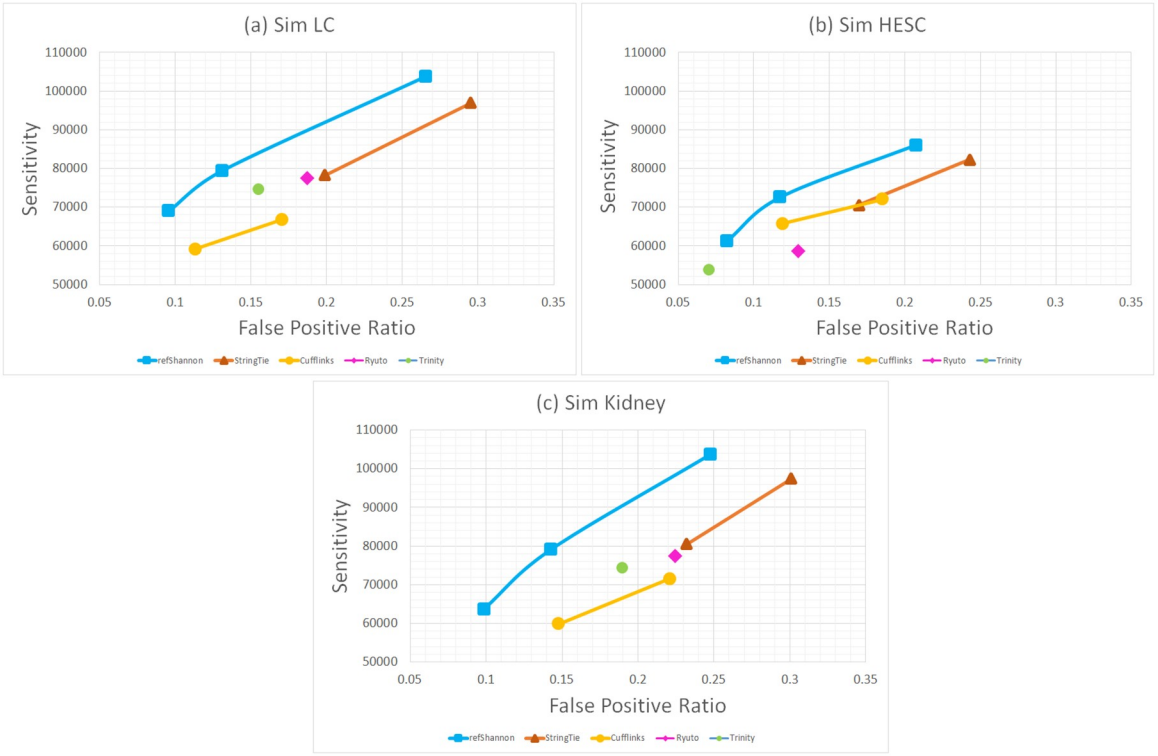

Fig 3. ROC analysis of simulated datasets.

https://doi.org/10.1371/journal.pone.0232946.g003

Cufflinks in default setting. Quantitatively, if we fix false positive ratio at 15.5%, we obtain a sensitivity gain of 13.5% over guided Trinity which has the second best ROC point. The conclusion is similar in the Kidney dataset, where RefShannon obtains a sensitivity gain of 22% over guided Trinity at 19% false positive ratio. In the HESC dataset, RefShannon shows an ROC trend similar to guided Trinity, and has a sensitivity gain of 10.6% over Cufflinks which shows the next best ROC point at 12% false positive ratio. There is a larger gain in LC and Kidney datasets of pair-end reads than in HESC dataset of single-end reads probably because RefShannon is able to better utilize the pair-end information not only in the splice graph construction but also in the flow decomposition stage.

Sensitivity of real datasets

For real datasets, since it's hard to judge if a reconstructed transcript is a false positive one or an unknown transcript yet to be discovered. Therefore, we focus only on sensitivity performance which means among the known reference transcripts, how many of them are correctly recovered.

The sensitivity calculation is similar to previous ROC approach. The difference is (1) For fair evaluation of sensitivity, we run assemblers all in their max sensitivity settings (detailed configurations in S4 File). (2) We only use the oracle set of reference transcripts (statistics in Table 1) for each real dataset. The oracle set contains reference transcripts that are fully covered by reads (tolerated by 25 bp from both ends). We expect these well expressed reference transcripts shall be reconstructed by assemblers.

Fig 4 shows our sensitivity evaluation for the three (i.e. LC, HESC, Kidney) real datasets. In the three subplots of first column, reference transcripts in the oracle set are grouped according to their read coverage. If a reference transcript has a low (or high) read coverage, it implies its expression level in cells is low (or high). Note the oracle sets of reference transcripts are different among three datasets, so the group values per dataset are different. For LC and Kidney datasets, most of the reference transcripts are within lower coverage. Therefore, we group the reference transcripts into read coverage of 60, 10, 10, 10 and 10 percentile. For HESC, the reference transcripts are grouped into 20 percentile each. For each dataset under various read coverage conditions, RefShannon has recovered reference transcripts better than all other assemblers. In the three subplots of second column, reference transcripts are grouped according to their isoform multiplicity. Recall that a gene may contain multiple transcripts (i.e. isoforms) due to alternative splicing, and the isoform multiplicity of a transcript refers to the number of isoforms of that transcript's gene. A higher isoform multiplicity value implies a more complex splice graph for recovery and also implies longer transcript length. Here the reference transcripts for HESC have relatively simple isoform multiplicity (most equals to 1), so we group reference transcripts into about 70, 15 and 15 percentile, while the reference transcripts for LC and Kidney datasets are grouped into 20 percentile each. For each dataset under various isoform multiplicity regions, RefShannon has also recovered more reference transcripts than all other assemblers. Note also that Cufflinks's performance drops in LC and Kidney compared to HESC, this could be because reads from LC and Kidney datasets are pair end and there are more read alignments Cufflinks may consider to be compatibility uncertain and thus throw away. In addition, guided Trinity does a good job to recover the transcripts of high expression levels in HESC dataset, this could be because the relevant underlying assembly graphs are less fragmented.

Computation resources

To understand how much time/memory RefShannon requires for assembly tasks, we have monitored assembly procedures using the cgmemtime tool (https://github.com/gsauthof/cgmemtime),

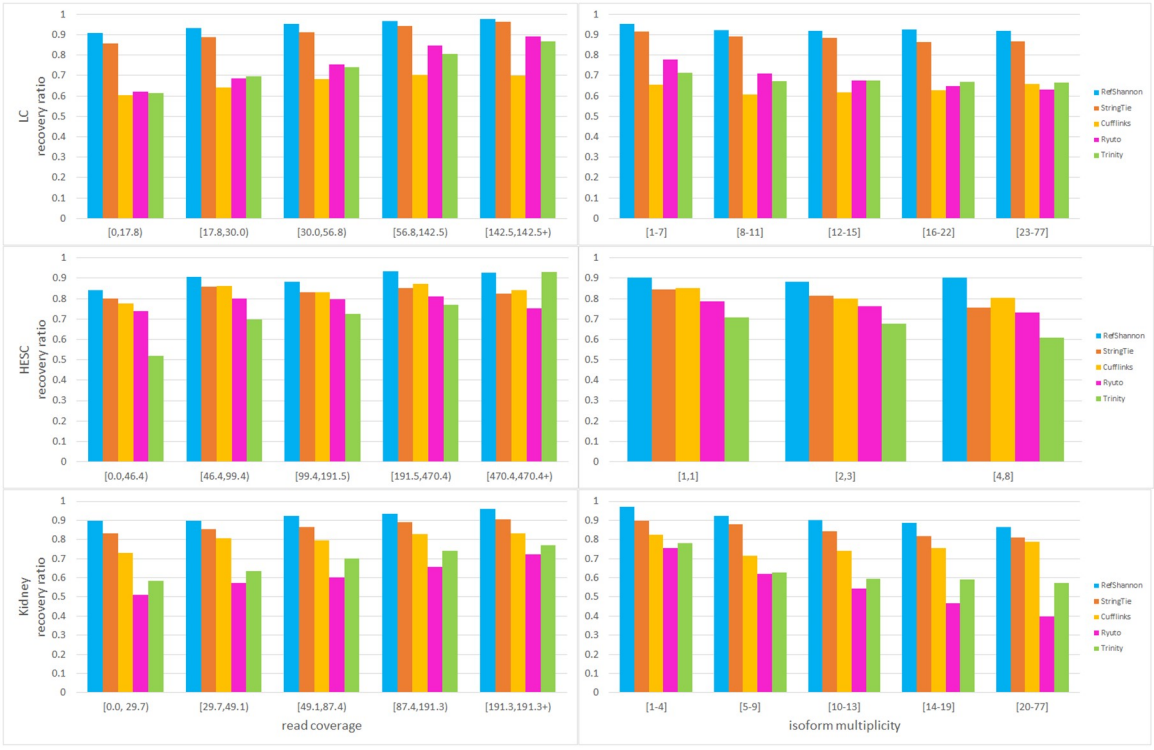

Fig 4. Sensitivity analysis of real datasets. The y-axis of each subplot is recovery ratio, the ratio of the number of correctly reconstructed reference transcripts over the total number of (oracle) reference transcripts. The x-axis of each subplot in left column is read coverage, and the x-axis of each subplot in right column is isoform multiplicity. A lower read coverage implies less expression level of reference transcript in cells, and a higher isoform multiplicity implies more complex splicing patterns. Isoform multiplicity in HESC is lower than that of LC data, implying a simpler splicing structure of reference transcripts in HESC data.

https://doi.org/10.1371/journal.pone.0232946.g004

which was previously adopted to compare the computational complexities among read aligners [34]. We put details in S7 File. RefShannon is overall faster than guided Trinity, Cufflinks, and Ryuto (for large dataset and more processes). RefShannon consumes more memory compared to other assemblers (except guided Trinity which essentially conducts de novo assembly). This could be because RefShannon is written in Python and memory sharing is less efficient especially for multiprocessing. Currently, a typical lab server with at least 20 CPU cores and over 200GB memory would be sufficient to run RefShannon on large real datasets. One of our future direction is to further improve its computational efficiency.

Discussions

We have developed RefShannon—a new genome-guided transcriptome assembler as an extension of the original de novo Shannon project [24]. It utilizes a careful splice graph generation procedure aimed at capturing as much information as possible from read alignments, and utilizes a sparse flow decomposition algorithm aims at reconstructing as small number of transcripts as possible under splice graph constraints. Our evaluation shows performance gain of RefShannon over state-of-art genome guided assemblers for both simulated and real datasets. We expect RefShannon will help discover novel genes and isoforms that may be missed by existing transcriptome assemblers. We also expect its intermediately generated splice graphs (with nodes, edges and paths) will provide helpful interface to be used by other relevant research.

There are several future directions. One of them is to further improve the computational efficiency as described previously.

In addition, it shall be useful to incorporate the third generation long reads (e.g. PacBio [35] and Nanopore reads [36]) to assist the short read assembly process. Compared to the short RNA reads (e.g. <300 bp), long reads (e.g. >10000 bp) will be able to better bridge multi-exons and resolve repetitive regions. However they are more expensive (e.g. 100 to 280 Euro per isolate) and have higher errors (typically 10% to 15%), whereas short reads are more cost-effective (e.g. 40 Euro per isolate) and also much more accurate (typically below 0.1% error rate) [37, 38]. Ideally, applying long contigs (or reads) into short read assemblers could be an effective feature [17, 20].

Moreover, it could be interesting to apply RefShannon onto the downstream analysis such as variant calling (e.g. SNP or small indels). RefShannon shows a better sensitivity (under a similar false positive rate) than existing assemblers, which implies additional new RNA transcripts could be discovered. Using variant callers (e.g. GATK [39] or recently developed abSNP [40]) to look for variants from newly discovered transcripts may help scientists gain new medical insights.

Methods

As the overall work flow is described in Results, in this section, we will describe the graph generation and traversal steps of RefShannon.

Splice graph generation

The splice graph generation consists of three steps: split, merge and connect. The pseudo code can be obtained from Algorithm 1 in S2 File.

In the split step (Fig 5B), we divide inferred exon regions (Fig 5A) into sub exon regions where splice junctions may occur. Splice junctions represent the exon or sub-exon boundaries

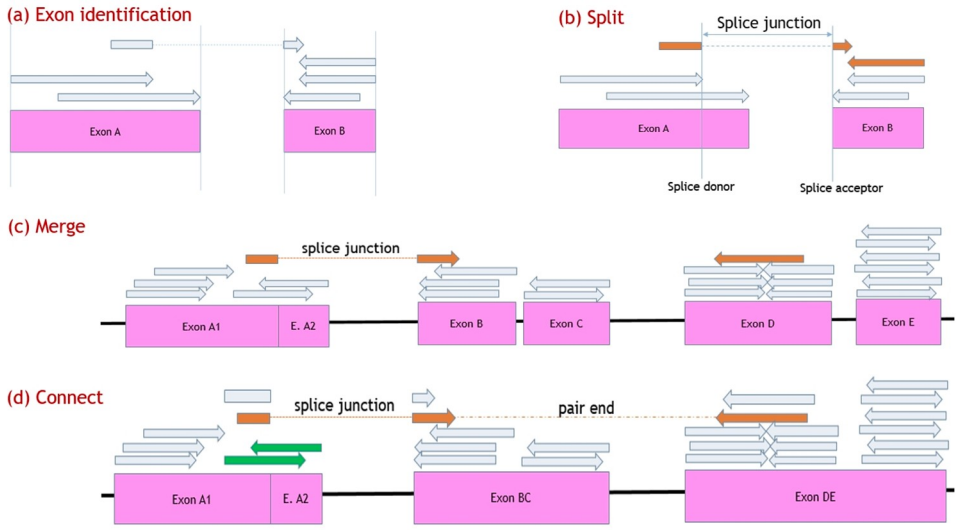

Fig 5. Illustrations of splice graph generation. (A) Exon regions are identified based on read alignments. (B) A splice event happens in the middle of Exon A, which implies Exon A should be further split into two sub exons at splice donor location. (C) Exon A1 and A2 should not be merged due to a splice event after A1. Exon B and C shall be merged if their gap is small. Exon D and E shall be merged if their gap is moderate but their coverage is high. (D) Exon A1 and A2 are connected because there're read alignments (green color) crossing them. Exon A1 and BC are connected because there're splice junctions (brown color) between them. Exon BC and DE are connected because they each contain one side of read pair alignment (brown color) and there are no other exon regions between them. A known path (A1, BC, DE) is collected because the read pair alignment (brown color) involves these three nodes.

https://doi.org/10.1371/journal.pone.0232946.g005

where alternative splicing could occur. They can be determined according to the read alignments, because a read sampled across two exon parts in a transcript can be split aligned onto disconnected genome regions, with the locus where the read leaves as the splice donor and the locus where the read enters as the splice acceptor. For example, a 100-bp read could be split-aligned onto the genome (chromosome 15) at loci [78837259, 78837318] and loci [78837519, 78837558], then we consider there is a splice junction at locus 78837318 (as splice donor) and at locus 78837519 (as splice acceptor). We determine a splice junction if there's at least one supporting alignment. This is a loose criteria and may bring false edges in the splice graph. However, this is alleviated by the next step of flow decomposition because such edges have low weights and usually will be ignored as there can be no flow decomposed to go through these edges. Once we determine a splice junction (splice donor or acceptor) in the middle of an exon, we need to divide the exon further into sub exons, because the splice junction indicates that a divided sub exon could link to another non-adjacent (sub) exon that may belong to the same RNA transcript.

In the merge step (Fig 5C), we empirically merge two exon regions based on their gap and expression levels in order to reduce the chances that the gap is actually part of exon but not covered by reads. For example, we merge two exon regions if they are very close to each other. Statistics have shown that less than 0.01% of introns are smaller than 20 bp in length [41], so a gap within 10 bp is highly unlikely intronic but should be an uncovered exonic region. We also merge two exon regions if they are moderately close to each other and have high read coverages. However, all merge procedures require two regions have no splice donors or acceptors between them, otherwise the genome gap between them should not occur together with the two regions in true transcripts.

In the connect step (Fig 5D), we establish weighted edges among nodes and collect known path information.

Two nodes are connected by a directed edge if there is a read alignment crossing one node to the other. The two nodes can be continuous, or discontinuous on the genome when there's a splice junction between them. The related edge weight is proportional to the number of crossing read alignments. Another situation to augment edges may occur only for pair end reads. As illustrated in Fig 5D, if a read pair alignment has its first segment onto one node and the second segment onto another node, and there're no other nodes between them, these two nodes should be adjacent in some true transcript and hereby an edge should be added to connect them, even if there're no read alignments crossing these two nodes. A similar idea has also occurred in Scripture [15]. The weight for the augmented edge here is proportional to the number of related read pair alignments.

A known path is a sequence of nodes inferred from read alignments. Known paths are collected when a read alignment crosses more than two nodes, or a read pair alignment for an augmented edge involves more than two nodes (also illustrated in Fig 5D). We store known paths as a tuple of triple nodes. This not only provides sufficient extra information for flow decomposition later, but also helps reduce memory.

Sparse flow decomposition

Sparse flow decomposition has been proposed in [24] for de novo assembly and here we modify it into the RefShannon framework for genome-guided assembly. Given a splice graph, it finds the minimum set of flows (e.g. paths with weights) that explains the splice graph.

Mathematically, given graph $G = (V, E)$, we need to find $\text{argmin}_T|T|$ such that $\forall v \in V$, $\forall e \in \text{InEdges}(v) \cup \text{OutEdges}(v)$, $w_e = \sum_{t \in T, e \in t} f(t)$ where $t \in T$ is a path in G corresponds to an RNA

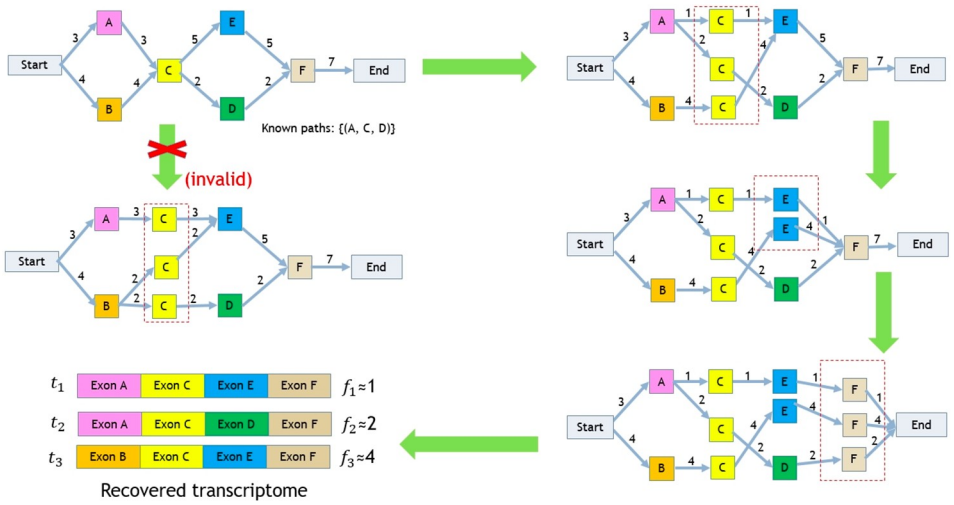

Fig 6. Example of sparse flow decomposition. In this figure, starting from the initial splice graph on the top left side, we try to recover transcriptome by doing local decomposition for node C,E and F iteratively.

https://doi.org/10.1371/journal.pone.0232946.g006

transcript and $f(t)$ represents the flow weight of t. This is a hard task, because there can be $|P| = 2^{|V|}$ paths, and thus $2^{|P|}$ possible sets of flows.

Instead, we try to decompose a set of flows out of the splice graph node by node in topological order, as illustrated in Fig 6 (For more details, please refer to Algorithm2 and Algorithm3 in S2 File). Specifically, we first add a special source node to connect all nodes without in-edges and a special sink node to connect all nodes without out-edges. We then do local sparse flow decomposition iteratively for each node. Finally we collect flows that start from source node and stop at sink node as reconstructed RNA transcripts.

The local sparse flow decomposition at node v tries to find the minimum number of flows through v restricted by edge weight constraints. It can be formally described as: find $\text{argmin}_f ||f||_0$ such that $\sum_{i \in \text{InEdges}(v)} f_{i,j} = w_j, \forall j \in \text{OutEdges}(v)$ and $\sum_{j \in \text{OutEdges}(v)} f_{i,j} = w_i, \forall i \in \text{InEdges}(v)$ and $f_{i,j} \geq 0$. Here $||f||_0$ means the support set of $\{f_{i,j}\}$, or the number of positive $f_{i,j}$s.

Solving this problem may not be practically efficient. Consider for each node v, we have $m \times n$ possible flows ($m = |\text{InEdges}(v)|$, $n = |\text{OutEdges}(v)|$). We need to enumerate all 2^{mn} path combinations to figure out which combination offers us min $||f||_0$ and keeps the edge weight constraints.

One possible thought to ease the problem, as commonly used in signal processing, is to approximate $||f||_0$ by $||f||_1$, which unfortunately turns out to be a constant: $||f||_1 = \sum_{i,j} f_{i,j} = \sum_i w_i = \sum_j w_j$. Meanwhile, we have noticed that $||f||_0 = ||f \odot r||_0$, $r = \{r_{i,j} | r_{i,j} > 0\}$($\odot$ is element-wise product). So instead of approximating $||f||_0$ by $||f||_1$, we try to approxiate $||f||_0$ by $||f \odot r||_1 = \sum_{i,j} f_{i,j} r_{i,j}$.

Therefore, we have relaxed the original problem as: find $\text{argmin}_f \sum_{i,j} f_{i,j} r_{i,j}$ such that $\sum_{i \in \text{InEdges}(v)} f_{i,j} = w_j, \forall j \in \text{OutEdges}(v)$ and $\sum_{j \in \text{OutEdges}(v)} f_{i,j} = w_i, \forall i \in \text{InEdges}(v)$ and $f_{i,j} \geq 0$, $r_{i,j} > 0$. We could use linear programming (e.g Python CVXOPT package (http://cvxopt.org/)) to solve the above problem. Since the result may contain noise, we also need to do some thresholding to get a sparse solution. To get the sparsest result, we generate r by a number of times and select the sparsest f as the final local sparse flow decomposition solution. An illustration of why the solution tends to be sparse is provided in Fig 3 in S2 File.

It is also possible that the local sparse flow decomposition may bring two results that have the same lowest sparsity and satisfy the edge constraints. This can be resolved if one of them includes a known path, as illustrated in Fig 6 as well as Fig 2C.

Supporting information

S1 File. Compare Shannon with StringTie.
(PDF)

S2 File. RefShannon algorithm details.
(PDF)

S3 File. Additional comparisons among different assemblers.
(PDF)

S4 File. Parameter setting for different assemblers.
(PDF)

S5 File. Different thresholds on sensitivity and false positive.
(PDF)

S6 File. Comparison of assembly performance (ROC) using different aligners.
(PDF)

S7 File. Compare memory and time consumption of RefShannon to other assemblers.
(PDF)

Acknowledgments

The authors would like to thank Joseph Hui and Kayvon Mazooji for their support at the initial stage of the project.

Author Contributions

Conceptualization: Lior Pachter, David Tse, Sreeram Kannan.

Data curation: Sreeram Kannan.

Formal analysis: Shunfu Mao.

Funding acquisition: Lior Pachter, David Tse, Sreeram Kannan.

Investigation: Shunfu Mao, Lior Pachter, David Tse, Sreeram Kannan.

Methodology: Shunfu Mao, Lior Pachter, David Tse, Sreeram Kannan.

Project administration: Lior Pachter, David Tse, Sreeram Kannan.

Software: Shunfu Mao, Sreeram Kannan.

Supervision: Sreeram Kannan.

Validation: Shunfu Mao.

Visualization: Shunfu Mao.

Writing – original draft: Shunfu Mao.

Writing – review & editing: Shunfu Mao, Sreeram Kannan.

References

1. Witkowski JA. The discovery of 'split' genes: a scientific revolution. Trends in Biochemical Sciences. 1988; 13(3):110–113. https://doi.org/10.1016/0968-0004(88)90052-7 PMID: 3072705
2. Michael D, Manyuan L. Intron–exon structures of eukaryotic model organisms. Nucleic Acids Research. 1999; 27(15):3219–3228. https://doi.org/10.1093/nar/27.15.3219

3. Gilbert W. Why genes in pieces? Nature. 1978; 271(5645):501–501. https://doi.org/10.1038/271501a0 PMID: 622185
4. Breitbart RE, Andreadis A, Nadal-Ginard B. Alternative Splicing: A Ubiquitous Mechanism for the Generation of Multiple Protein Isoforms from Single Genes. Annual Review of Biochemistry. 1987; 56(1):467–495. https://doi.org/10.1146/annurev.bi.56.070187.002343 PMID: 3304142
5. Black DL. Mechanisms of Alternative Pre-Messenger RNA Splicing. Annual Review of Biochemistry. 2003; 72(1):291–336. https://doi.org/10.1146/annurev.biochem.72.121801.161720 PMID: 12626338
6. Meister G. RNA Biology: An Introduction. Wiley-VCH; 2011.
7. Wang Z, Gerstein M, Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics. 2009; 10(1):57–63. https://doi.org/10.1038/nrg2484 PMID: 19015660
8. Martin JA, Wang Z. Next-generation transcriptome assembly. Nature Reviews Genetics. 2011; 12 (10):671–682. https://doi.org/10.1038/nrg3068 PMID: 21897427
9. Alberts B, Hopkin K, Johnson AD, Morgan D, Raff M, Roberts K, et al. Essential Cell Biology (Fifth Edition). W. W. Norton & Company; 2019.
10. Leonelli S, Ankeny RA. What makes a model organism? Endeavour. 2013; 37(4):209–212. https://doi.org/10.1016/j.endeavour.2013.06.001 PMID: 23849606
11. Robertson G, Schein J, Chiu R, Corbett R, Field M, Jackman SD, et al. De novo assembly and analysis of RNA-seq data. Nature Methods. 2010; 7(11):909–912. https://doi.org/10.1038/nmeth.1517 PMID: 20935650
12. Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nature Biotechnology. 2011; 29(7):644–652. https://doi.org/10.1038/nbt.1883 PMID: 21572440
13. Schulz MH, Zerbino DR, Vingron M, Birney E. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. Bioinformatics. 2012; 28(8):1086–1092. https://doi.org/10.1093/bioinformatics/bts094 PMID: 22368243
14. Xie Y, Wu G, Tang J, Luo R, Patterson J, Liu S, et al. SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads. Bioinformatics. 2014; 30(12):1660–1666. https://doi.org/10.1093/bioinformatics/btu077 PMID: 24532719
15. Guttman MTcioAcR, Garber M, Levin JZ, Donaghey J, Robinson J, Adiconis X, et al. Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. Nat Biotech. 2010; 28(5):503–510. https://doi.org/10.1038/nbt.1633
16. Trapnell C, Williams BA, Pertea G, Mortazavi A, Kwan G, van Baren MJ, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat Biotechnol. 2010; 28(5):511–515. https://doi.org/10.1038/nbt.1621 PMID: 20436464
17. Pertea M, Pertea GM, Antonescu CM, Chang TC, Mendell JT, Salzberg SL. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. Nat Biotech. 2015; 33:290–295. https://doi.org/10.1038/nbt.3122
18. Liu J, Yu T, Jiang T, Li G. TransComb: genome-guided transcriptome assembly via combing junctions in splicing graphs. Genome Biology. 2016; 17(1). https://doi.org/10.1186/s13059-016-1074-1
19. Song L, Sabunciyan S, Florea L. CLASS2: accurate and efficient splice variant annotation from RNA-seq reads. Nucleic Acids Research. 2016; 44(10):e98–e98. https://doi.org/10.1093/nar/gkw158 PMID: 26975657
20. Gatter T, Stadler PF. Ryūtō: network-flow based transcriptome reconstruction. BMC Bioinformatics. 2019; 20(1). https://doi.org/10.1186/s12859-019-2786-5 PMID: 30991937
21. Liu R, Dickerson J. Strawberry: Fast and accurate genome-guided transcript reconstruction and quantification from RNA-Seq. PLOS Computational Biology. 2017; 13(11):e1005851. https://doi.org/10.1371/journal.pcbi.1005851 PMID: 29176847
22. Steijger T, Abril JF, Engstrom PG, Kokocinski F, Consortium TR, Hubbard TJ, et al. Assessment of transcript reconstruction methods for RNA-seq. Nat Meth. 2013; 10(12):1177–1184. https://doi.org/10.1038/nmeth.2714
23. Hayer KE, Pizarro A, Lahens NF, Hogenesch JB, Grant GR. Benchmark analysis of algorithms for determining and quantifying full-length mRNA splice forms from RNA-seq data. Bioinformatics. 2015; p. btv488. https://doi.org/10.1093/bioinformatics/btv488 PMID: 26338770
24. Kannan S, Hui J, Mazooji K, Pachter L, Tse D. Shannon: An Information-Optimal de Novo RNA-Seq Assembler. bioRxiv. 2016.
25. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, et al. STAR: ultrafast universal RNA-seq aligner. Bioinformatics. 2012. https://doi.org/10.1093/bioinformatics/bts635 PMID: 23104886

26. Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, Salzberg SL. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. Genome Biology. 2013; 14(4): R36. https://doi.org/10.1186/gb-2013-14-4-r36 PMID: 23618408

27. Kim D, Langmead B, Salzberg SL. HISAT: a fast spliced aligner with low memory requirements. Nat Meth. 2015; 12(4):357–360. https://doi.org/10.1038/nmeth.3317

28. Wu TD, Watanabe CK. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. Bioinformatics. 2005; 21(9):1859–1875. https://doi.org/10.1093/bioinformatics/bti310 PMID: 15728110

29. Li H. Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics. 2018; 34(18):3094–3100. https://doi.org/10.1093/bioinformatics/bty191 PMID: 29750242

30. Zimin AV, Marcais G, Puiu D, Roberts M, Salzberg SL, Yorke JA. The MaSuRCA genome assembler. Bioinformatics. 2013; 29(21):2669–2677. https://doi.org/10.1093/bioinformatics/btt476 PMID: 23990416

31. Au KF, Sebastiano V, Afshar PT, Durruthy JD, Lee L, Williams BA, et al. Characterization of the human ESC transcriptome by hybrid sequencing. Proc Natl Acad Sci USA. 2013; 110(50):E4821–4830. https://doi.org/10.1073/pnas.1320101110 PMID: 24282307

32. Tilgner H, Grubert F, Sharon D, Snyder MP. Defining a personal, allele-specific, and single-molecule long-read transcriptome. Proc Natl Acad Sci USA. 2014; 111(27):9869–9874. https://doi.org/10.1073/pnas.1400447111 PMID: 24961374

33. Li B, Dewey CN. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics. 2011; 12(1):323. https://doi.org/10.1186/1471-2105-12-323 PMID: 21816040

34. Križanović K, Echchiki A, Roux J, Šikić M. Evaluation of tools for long read RNA-seq splice-aware alignment. Bioinformatics. 2017; 34(5):748–754.

35. Roberts RJ, Carneiro MO, Schatz MC. The advantages of SMRT sequencing. Genome Biology. 2013; 14(6). https://doi.org/10.1186/gb-2013-14-7-405 PMID: 23822731

36. Maitra RD, Kim J, Dunbar WB. Recent advances in nanopore sequencing. ELECTROPHORESIS. 2012; 33(23):3418–3428. https://doi.org/10.1002/elps.201200272 PMID: 23138639

37. The long view on sequencing. Nature Biotechnology. 2018; 36(4):287–287. https://doi.org/10.1038/nbt.4125 PMID: 29621212

38. Maio ND, Shaw LP, Hubbard A, George S, Sanderson ND, Swann J, et al. Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes. Microbial Genomics. 2019; 5(9). https://doi.org/10.1099/mgen.0.000294 PMID: 31483244

39. DePristo MA, Banks E, Poplin R, Garimella KV, Maguire JR, Hartl C, et al. A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nature Genetics. 2011; 43(5):491–498. https://doi.org/10.1038/ng.806 PMID: 21478889

40. Mao S, Mohajer S, Ramachandran K, Tse D, Kannan S. abSNP: RNA-Seq SNP Calling in Repetitive Regions via Abundance Estimation. In: Schwartz R, Reinert K, editors. 17th International Workshop on Algorithms in Bioinformatics (WABI 2017). vol. 88 of Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik; 2017. p. 15:1–15:14. Available from: http://drops.dagstuhl.de/opus/volltexte/2017/7658.

41. Sakharkar MK KP Chow VT. Distributions of exons and introns in the human genome. In Silico Biol. 2004; 4:387–93.