Accurate assembly of multi-end RNA-seq data with Scallop2

Qimin Zhang[1], Qian Shi[1], and Mingfu Shao[1,2,*]

[1]Department of Computer Science and Engineering, School of Electronic Engineering and Computer Science, The Pennsylvania State University

[2]Huck Institutes of the Life Sciences, The Pennsylvania State University

[*]correspondence should be addressed to: mxs2589@psu.edu

May 23, 2022

Modern RNA-sequencing protocols can produce multi-end data, where multiple reads originating from the same transcript are attached to the same barcode. The long-range information in the multi-end reads is beneficial in phasing complicated spliced isoforms, but assembly algorithms that leverage such information are lacking. Here we introduce Scallop2, a reference-based assembler optimized for multi-end RNA-seq data. The algorithmic core of Scallop2 consists of three steps: (1) using an algorithm to "bridge" multi-end reads into single-end phasing paths in the context of a splice graph, (2) employing a method to refine erroneous splice graphs by utilizing multi-end reads that fail to bridge, and (3) piping the refined splice graph and the bridged phasing paths into an algorithm that integrates multiple phase-preserving decompositions. Tested on 561 cells in two Smart-seq3 datasets and on 10 Illumina paired-end RNA-seq samples, Scallop2 substantially improves the assembly accuracy compared to two popular assemblers StringTie2 and Scallop.

Main

The established high-throughput RNA sequencing technologies (RNA-seq) have fueled many biological and biomedical discoveries. One of the critical steps and yet the major computational challenges in RNA-seq analysis is *transcript assembly*—the reconstruction of full-length expressed transcripts captured by a given RNA-seq sample. Over the decades, tremendous efforts have been made to develop efficient approaches for transcript assembly, including these for short-read RNA-seq data (e.g., Cufflinks[1], Scripture[2], Traph[3], CLASS2[4], TransComb[5], Scallop[6], StringTie[7], StringTie2[8], and RefShannon[9]), and these for long-read RNA-seq data (e.g., Scallop-LR[10] and StringTie2[8]). Theoretical and algorithmic advances have also been made in studying the flow-decomposition problems and their variants abstracted from transcript assembly[11,12,13].

Despite these, the task of transcript assembly is far from being solved. Assemblers for short-read RNA-seq data have been struggling with correctly phasing the complete intron-chains from (short) reads especially when genes expressing numerous alternatively spliced isoforms. It remains algorithmically challenging to efficiently use the paired-end information to resolve such complicated cases. Long-read RNA-seq data has its own limitation of low coverage, making it less efficient in identifying lowly expressed transcripts. A large portion of long reads remain transcript fragments due to complementary DNA (cDNA) synthesis and sequencing length limit[14], and hence the assembly of long-reads data faces the same challenge as short-reads data. According to benchmarking studies[15] and recent assemblers, the accuracy of existing methods remains unsatisfactory; improved algorithms are therefore still urged for transcript assembly.

The recently developed protocols such as Smart-seq series[16,17] provide single-cell RNA-seq data with coverage spanning full-length molecules (as opposed to protocols that only captures 3' or 5' end of mRNAs), which therefore can be used to assemble transcriptomes at single-cell resolution. Smart-seq3 employed a barcoding technology that can produce *multi-end* RNA-seq data, for which (a subset of) reads originating from the same transcript will be attached to the same barcode. Such multi-end RNA-seq data offers strong long-range phasing constraints that are promising for resolving complicated splicing variants, but again they are inherently difficult to use, as there often exist gaps (which might contain missing junctions) between consecutive ends (a similar situation with paired-end RNA-seq data).

We present Scallop2, a reference-based transcript assembler optimized for multi-end and paired-end RNA-

seq data. To make use of the long-range constraints in multi-end reads, Scallop2 proposes an algorithm that infers a single, long path in the splice graph that connects all ends attached with the same barcode. This algorithm solves a formulation that calculates a path with maximized bottleneck-weight, which characterizes the true bridging path while also robust to sequencing/alignment errors and transcript noises. Second, it's long been recognized that the quality of the underlying splice graph highly affects the assembly accuracy. Scallop2 uses the multi-end reads that are failed to bridge, which are often due to incomplete splice graphs, to determine the false starting and ending vertices and consequently to refine the splice graph. Third, the central formulation of transcript assembly seeks a decomposition of the (refined) splice graph with the (bridged) long-range constraints. Scallop2 proposes an improved algorithm for this formulation by sampling multiple "phase-preserving" decompositions followed by a consensus approach to select transcripts. These algorithmic advances of Scallop2 lead to its high accuracy in assembling multi-end RNA-seq data (e.g., Smart-seq3 data). Substantial improvement is also obtained in assembling paired-end RNA-seq data, a special case of multi-end RNA-seq data with exactly two ends in each read group.

We compare Scallop2 with three recent transcript assemblers, StringTie2[8], Scallop[6], and CLASS2[4], on both Smart-seq3 data (including a human dataset named HEK293T with 192 cells and a mouse dataset named Mouse-Fibroblast with 369 cells) and Illumina paired-end RNA-seq data (including 10 samples named ENCODE10). Their assembly accuracies are evaluated using gffcompare[18]. Please refer to Supplementary Notes 1–2 and Supplementary Figure 15 for more details about methods compared, accuracy measures, and evaluation pipeline.

Scallop2 achieves both the highest sensitivity and precision on the two Smart-seq3 datasets (**Figure 1a**). More specifically, Scallop2 produces 3.6%, 18.5%, and 24.3% more matching transcripts (proportional to sensitivity) than StringTie2, Scallop, and CLASS2 averaged over all cells in the HEK293T dataset, and assembles 1.7%, 13.4%, and 20.7% more matching transcripts than them on the Mouse-Fibroblast dataset. In terms of precision, Scallop2 improves 93.6%, 34.2% and 105.5% comparing with StringTie2, Scallop, and CLASS2 on the HEK293T dataset, and improves 75.9%, 19.3%, and 62.7% than them on the Mouse-Fibroblast dataset. A scatter plot that integrates precision and sensitivity is given in Supplementary Figure 1. An aligned comparison at the level of individual cells separately for precision and sensitivity is given in Supplementary Figure 2.

One method might obtain higher (respectively lower) precision but lower (respectively higher) sensitivity.

To quantitatively compare two methods in this case, we calculate the *adjusted precision*, defined as their precisions when their sensitivities are adjusted to the same (see Supplementary Note 3). For the Smart-seq3 data (**Figure 1b**), Scallop2 improves 97.6%, 65.6%, and 161.1% in terms of adjusted precision than StringTie2, Scallop, and CLASS2 on the HEK293T dataset, and improves 79.6%, 36.8%, and 94.9% comparing with these three methods respectively on the Mouse-Fibroblast dataset. An aligned comparison in adjusted precision at the level of individual cells is illustrated in Supplementary Figure 3.

Scallop2 also achieves the highest sensitivity and the highest precision on Illumina paired-end RNA-seq data (**Figure 1c**). Averaged over the ENCODE10 data, Scallop2 produces 23.4%, 9.6% and 43.3% more matching transcripts than StringTie2, Scallop and CLASS2 when these samples are aligned with HISAT2, and produces 25.2%, 6.5% and 40.2% more matching transcripts than them with STAR. In terms of precision, Scallop2 improves 1.1%, 4.1% and 31.1% comparing with StringTie2, Scallop and CLASS2 with HISAT2, and improves 4.1%, 4.3% and 15.6% than them with STAR.

When comparing Scallop2 with each of the other three methods at the same level of sensitivity (**Figure 1d**), Scallop2 improves 46.6%, 27.4%, and 105.7% in adjusted precision than StringTie2, Scallop, and CLASS2 averaged over the ENCODE10 data aligned by HISAT2, and improves 50.9%, 20.0%, and 89.6% than the three methods respectively when aligned with STAR.

A comparison at individual samples is given in **Figure 1e**, with the precision-sensitivity curves for Scallop2 illustrated. (The curves are drawn by gradually filtering out the lowly-expressed transcripts assembled by Scallop2; see Supplementary Note 3.) These curves always lie to the right of the points corresponding to other methods, suggesting that Scallop2 always achieves higher precision at the same level of sensitivity.

The CPU time and peak memory of the 4 methods is reported in Supplementary Table 1 and Supplementary Table 2. StringTie runs the fastest on Illumina data and on the Mouse-Fibroblast dataset; Scallop leads on the HEK293T dataset. In memory footprint, StringTie2 significantly outperforms other methods in all cases. We argue that Scallop2 runs reasonably fast and uses acceptable memory. For example, on average over the ENCODE10 data aligned with HISAT2, Scallop2 takes about 60 minutes (while StringTie2 takes about 8 minutes) and uses about 6GB memory (while StringTie2 uses 148MB).

Above experimental results evaluate all assembled transcripts, including single-exon transcripts. The accuracy solely about multi-exon transcripts are illustrated in Supplementary Figures 4–7: similar improvement

of Scallop2 over other methods are observed in assembling multi-exon transcripts. To compare these methods in assembling different types of transcripts, we illustrate their accuracies in assembling transcripts with different numbers of exons (Supplementary Figure 8 and Supplementary Note 4) and transcripts expressed at different levels (Supplementary Figure 9 and Supplementary Note 5). We then evaluate their accuracies with varying parameters (Supplementary Figures 10–12 and Supplementary Note 6). Finally, to illustrate the effects of the heuristics used in Scallop2, we describe how the heuristics lead to the correct assembly with examples (Supplementary Figure 13 and Supplementary Note 7); we also compare the accuracy of full-length transcripts and non-full-length transcripts assembled by Scallop2 to demonstrate the effectiveness of determining false vertices (Supplementary Figure 14 and Supplementary Note 8).

Scallop2 can be further improved in accuracy and running time. Current version of Scallop2 does not distinguish reads sampled from the exact locations and PCR duplicates, but we will explore statistical models that account for PCR duplicates and coverage bias, which will lead to more accurate estimation of weights of vertices and edges of the splice graph. Scallop2 is slower largely due to the consensus approach that decomposes the splice graph multiple times. We observe that a substantial number of vertices will be decomposed the same in multiple runs; we therefore will experiment algorithms that only decompose "volatile" vertices in different runs to speedup Scallop2.

Methods

We focus on reference-based transcript assembly. The RNA-seq reads will be first aligned to the reference genome using splice-aware aligners such as TopHat[19], STAR[20], or HISAT2[21]. Scallop2 takes the reads alignment, in standard sam/bam format, as input. Scallop2 is able to assemble both strand-specific and non-strand-specific RNA-seq data. Users can either specify the "library-type" of the protocol (*first* or *second*, corresponding to *fr-firststrand* or *fr-secondstrand* options in TopHat, or *unstreanded*), or Scallop2 can automatically detect the library type based on the *XS* tag available in the alignment. Given the reads alignment, Scallop2 first assigns all aligned reads to either the forward strand or the reverse strand. If the protocol is strand-specific, Scallop2 can infer which strand (i.e., either forward strand or the reverse strand) a read originates from by using the 0x10, 0x20, 0x40, and 0x80 bits of FLAG field available in the reads alignment together with the given or inferred library type. If the protocol is not strand-specific (i.e., *unstranded*), the originating strand of reads that contain junctions are usually inferred by the aligner based on the canonical splicing patterns and available to Scallop2 through the *XS* tag; reads without junctions will be assigned to the strand with existing overlapping reads that contain junctions. After assigning reads to strands, reads on the same strand and overlapping in genomic coordinates are then partitioned as *gene loci*. Individual gene locus will be assembled independently. The assembled transcripts will be written into a file in the standard gtf format, where a "+" or "-" will be coped with each assembled transcript to indicate its originating strand ("+" for forward strand and "-" for reverse strand).

Below, we describe the data structures we use to organize all splicing signals and multi-end constraints in the alignments, followed by the formulation of the multi-end assembly problem and our algorithm for solving this formulation. Prior to these, we first elucidate the difference between Scallop2 and Scallop. In constructing the data structures, Scallop2 uses the same weighted splice graph as in Scallop, but Scallop2 represents the long-range signals in the data as multi-end phasing paths while Scallop represents such information as (single-end) phasing paths. The problem formulation used in Scallop2 (i.e., multi-end transcript assembly) extends the formulation of Scallop; the difference is that Scallop2 requires to preserve all multi-end phasing paths while Scallop preserves all (single-end) phasing paths. The algorithmic framework and the three components are all new to Scallop2, which also constitute the major algorithmic innovations of Scallop2.

Constructing the splice graph and multi-end phasing paths

For each gene locus, we build a weighted splice graph $G = (V, E, w)$ to organize the inferred exons and junctions in the alignments and a set of multi-end phasing paths $\mathcal{C}$ to keep the long-range information in multi-end read (see Extended Data Figure 1). Note that the construction of the weighted splice graphs in Scallop2 is the same as that in Scallop. The splicing positions are first extracted from the junctions (marked with 'N' in the CIGAR string of alignments) to obtain the boundaries of exons (or partial exons) and introns of the reference genome. For each inferred exon, we add a vertex v to vertex-set V. A directed edge $e = (u, v)$ is added to edge-set E when there exists reads connecting exons u and v, where u occurs before v in the reference genome. We note that all splice graphs, no matter which strand this gene locus originates from, are constructed w.r.t. the forward strand (i.e., from 5' to 3'): edges are always from left to right, i.e., from low genomic coordinates to high coordinates of the reference genome. (Again the strandness of the gene loci are available and will be revealed when writting the assembled transcripts into gtf file.) The weight of edge e, denoted as $w(e)$, is calculated as the number of reads that connect u and v. Additionally, we add a source vertex s and sink vertex t to V; for each vertex $u \in V \setminus \{s, t\}$ with in-degree of 0 (called *starting* vertices), we add a directed edge (s, u) with weight $w(s, u) = \sum_{(u,v)\in E} w(u, v)$, and for each vertex $v \in V \setminus \{s, t\}$ with out-degree of 0 (called *ending* vertices), we add a directed edge (v, t) with weight $w(v, t) = \sum_{(u,v)\in E} w(u, v)$.

With the weighted splice graph $G = (V, E, w)$ being constructed, each read r can be represented as a path in G, denoted as $l(r)$, which is the list of vertices of G where r is aligned to. Let R be a set of reads from the same transcript/fragment; we call reads in R form a *read group*. In case of barcoding-based protocols like Smart-seq3, reads in group R include those attached with the same barcode; in case of paired-end RNA-seq protocols, the two ends of a pair form a group (i.e., $|R| = 2$); in case a read is not attached with any barcode, or its mate end is not aligned, then this read forms a group of its own. Notice that different reads in R might be aligned to the same path in G, i.e., $l(r_1) = l(r_2)$ for two reads $r_1, r_2 \in R$; for example in Extended Data Figure 1, the 3rd read (in red) in the 2nd row and the 2nd read (in purple) in the 4th row are aligned to the same path of $(2, 4)$. We define the *multi-end phasing path* obtained from R, denoted as $C(R)$, as the set of distinct paths collected from reads in R. Formally, $C(R) := \{l(r) \mid r \in R\}$. Let $\{R_1, R_2, \cdots, R_k\}$ be the collection of read groups in a gene locus. Again different groups of reads may correspond to the same multi-end phasing path, i.e., $C(R_i) = C(R_j)$ for some $1 \leq i < j \leq k$. We denote by $\mathcal{C}$ as the collection of distinct multi-end phasing paths in $\{C(R_1), C(R_2), \cdots, C(R_k)\}$. We note that Scallop represents reads as

single-end phasing paths, i.e., $\{l(r) \mid r \in R_1 \cup R_2 \cup \cdots \cup R_k\}$; it is in Scallop2 that we explicitly introduce multi-end phasing paths for representing long-range information in multi-end and paired-end RNA-seq data. To further elucidate these terms, a multi-end phasing path refers to multiple pieces of a single (unknown) path with possibly missing portions between consecutive (known) pieces. A single-end phasing path, on the other hand, refer to a (regular) path in the splice graph, which can either be obtained from the alignment of a single read, or from after filling up all missing portions of a multi-end phasing path (i.e., after bridging).

Problem formulation

For a gene locus, the above constructed weighted splice graph G together with the multi-end phasing paths $\mathcal{C}$ give a complete representation of the splicing information in the alignments and the long-range phasing information in the multi-end reads. The formulation takes G and $\mathcal{C}$ as input, and generates a set of *s-t* paths P of G and assigns an abundance $f(p)$ for each *s-t* path $p \in P$: each *s-t* path p infers an expressed transcript in this gene locus, and $f(p)$ predicts its expression abundance. We note that the input and output stated here are the same with the formulation in Scallop, except that here the set of multi-end phasing paths is part of the input while in Scallop it is the set of single-end phasing paths.

The optimization objectives of Scallop2 formulation also extend those used in Scallop. One key principle used in both formulations is to fully preserve the long-range information in the alignment. This principle is interpreted preserving all single-end phasing paths in Scallop but preserving all multi-end phasing paths in Scallop2; this constitutes the difference between the two formulations. First, since each multi-end phasing path is constructed from reads sampled from a single transcript, we expect that each multi-end phasing path appears in one of the reconstructed transcripts (i.e., in P). Formally, we say a multi-end phasing path $C \in \mathcal{C}$ is *covered* by P, if there exists an *s-t* path $p \in P$ such that every path $l \in C$ appears in p. We require all multi-end phasing paths in $\mathcal{C}$ be covered by P. Second, for each edge $e \in E$, we expect the abundance of the inferred *s-t* paths passing through e, i.e., $\sum_{p \in P: e \in p} f(p)$, is as close to its observed read coverage $w(e)$ as possible, which can be defined as to minimize the sum of the deviation between $w(e)$ and $\sum_{p \in P: e \in p} f(p)$ for all edges, denoted as $d(P, f) := \sum_{e \in E} |w(e) - \sum_{p \in P: e \in p} f(p)|$. Third, we aim to minimize $|P|$ following the parsimony principle.

Combining all the three objectives and the input and output, we informally describe the task of transcript assembly for multi-end RNA-seq data as follows.

Problem 1 (multi-end transcript assembly) *Given weighted splice graph $G = (V, E, w)$ and associated multi-end phasing paths $\mathcal{C}$, to compute a set of s-t paths P of G and abundance $f(p)$ for each $p \in P$, such that P covers all multi-end phasing paths in $\mathcal{C}$, and that both $d(P, f)$ and $|P|$ are as small as possible.*

Algorithmic framework

We propose a heuristic for above Problem 1, consisting of three steps, described below. This framework and the three techniques are particularly designed to improve assembly accuracy by fully leveraging the multi-end phasing information.

Step 1: bridging multi-end phasing paths into single-end phasing paths. Let $C = \{l_1, l_2, \cdots, l_n\} \in \mathcal{C}$ be a multi-end phasing path. All paths in C can be regarded as being sampled from a single (unknown) *s-t* path of the splice graph, representing the true transcript from which the reads used to construct C are generated. We propose to "bridge" all paths in C as a single-end phasing path, denoted as $h(C)$, in the splice graph. This task amounts to filling the gaps (if any) between consecutive paths in C. The algorithm to infer the true path that connects a pair of consecutive paths is one major innovation in Scallop2.

Step 2: determining false starting/ending vertices. We observe that failing to bridge a multi-end phasing path is mainly because the underlying splice graph is incomplete, such as certain junctions are missing due to low coverage or alignment errors. We design an algorithm to determine false starting and ending vertices in the splice graph (separately described below), which therefore improves the quality of the splice graph.

Step 3: decomposing splice graph. Let $H := \{h(C) \mid C \in \mathcal{C}\}$ be the set of bridged single-end phasing paths returned by Step 1. Let G' be the refined splice graph returned by Step 2. We propose an algorithm for Problem 1 by decomposing G' while preserving all bridged phasing paths in H. This improved graph-decomposition algorithm, separately described below, is the third algorithmic innovation of Scallop2.

Bridging algorithm

Let $C = \{l_1, l_2, \cdots, l_n\} \in \mathcal{C}$ be a multi-end phasing path. We sort all paths $\{l_1, l_2, \cdots, l_n\}$ in C in lexicographical order w.r.t. a topological ordering of all vertices of the splice graph (recall that the splice graph is a directed acyclic graph). See Extended Data Figure 2. We still write $C = (l_1, l_2, \cdots, l_n)$ after sorting all the paths lexicographically. We aim to "bridge" consecutive pairs of paths in C. If l_i and l_{i+1} overlap, i.e., there exists a suffix of l_i that is also a prefix of l_{i+1}, then l_i and l_{i+1} can be naturally merged into a single

path; otherwise, we will need to fill the gap by inferring a path connecting l_i to l_{i+1} in the splice graph. After all $n-1$ pairs of consecutive paths in C, i.e., $(l_1,l_2),(l_2,l_3),\cdots,(l_{n-1},l_n)$, are bridged, we can then connect them together as a single-end phasing path, i.e., $h(C)$. Below we focus on bridging non-overlapping pairs of paths.

Let $l=(a_1,a_2,\cdots,a_m),l'=(b_1,b_2,\cdots,b_n)$ be any two consecutive, non-overlapping paths in a multi-end phasing path. The problem of bridging l_i and l_{i+1} amounts to find a path in the splice graph G from a_m to b_1. Such path, called *bridging path*, infers the missing portion between l_i and l_{i+1}. We note that there might be multiple bridging paths due to alternative splicing and sequencing/alignment errors and we aim to infer the correct one. Below we first formulate the task of inferring the true bridging path as a new optimization problem and then design an efficient algorithm.

Formulation. We explore what is a good formulation to find the true bridging path. The main signal we have is the coverage information, and intuitively a bridging path supported by most reads are most likely the true path. We determined that, a formulation that seeks a bridging path that maximizes *bottleneck* weight, is suitable for this bridging problem. Below we formally describe this formulation.

We define a full ordering of all bridging paths. Let q_1 and q_2 be two arbitrary paths from a_m to b_1 in G. Let w_1^j (respectively w_2^j) be the jth smallest weight in path q_1 (respectively q_2). We say q_1 is *more reliable* than q_2, if there exists an integer k such that $w_1^j=w_2^j$ for all $1\leq j<k$, and $w_1^k>w_2^k$. In other words, each bridging path is represented as the *sorted* list of its weights in ascending order, and all bridging paths are then (implicitly) sorted in lexicographical order. We formulate the problem of bridging as to find the most reliable path. Intuitively, we seek a path q from a_m to b_1 in G such that the smallest weight in it is maximized; in case there are multiple paths with maximized smallest weight, among them we further seek the one whose second smallest weight is maximized, and so on.

We believe this formulation is appropriate for RNA-seq bridging. First, by maximizing bottleneck weight, the bridging paths are supported strongest, and hence are more likely to be the true bridging path. Second, through this formulation, false paths which are usually due to sequencing/alignment errors and therefore exhibit low abundances, can be automatically and efficiently excluded.

We note that this formulation satisfies the *optimal substructure* property: if $a_m\rightarrow u_1\rightarrow u_2\rightarrow\cdots\rightarrow u_l\rightarrow b_1$ is the most reliable path from a_m to b_1, then $a_m\rightarrow u_1\rightarrow u_2\rightarrow\cdots\rightarrow u_l$ is the most reliable path from a_m to

u_l. This allows us to design a dynamic programming algorithm to find the most reliable bridging path.

Dynamic programming algorithm. Let $(v_1, v_2, \cdots, v_{|V|})$ be a topological sorting of all vertices of the splice graph G (this is possible because splice graph is a directed acyclic graph). Given a particular v_i, we can use a single run to find the most reliable paths from v_i to v_j for every $j > i$. To compute the optimal path for v_j, we examine all vertices v_k that directly connects to v_j, and compare all paths stored in these vertices (each v_k already stores the most reliable paths from v_i to v_k at this time point). The optimal one will be kept and after concatenating v_j they become the most reliable paths from v_i to v_j. We run this subroutine for all v_i, $1 \leq i \leq |V|$, which gives the most reliable paths for all pairs of vertices in G. The overall running time of this algorithm is $O(|V|^2 \cdot |E|)$. To speed up, instead of maintaining the full list of the edge abundances for each path, whose length is $O(|V|)$, we only store the smallest M edge abundances (M is a parameter with default value of 5). This gives an improved running time of $O(M \cdot |V| \cdot |E|)$. Although optimality may not be guaranteed, experimental studies show that this heuristic rarely affects the overall accuracy.

Determining whether or not to accept the most reliable path. In case of paired-end RNA-seq data, the distribution of fragment length provides another source of information to decide if the inferred path is correct. Scallop2 will estimate the distribution of fragment length by sampling trivially-bridged reads (i.e., they overlap in the splice graph). Once we determine the most reliable path connecting l_1 and l_2 in a (paired-end) phasing path $C = (l_1, l_2)$, we can calculate the fragment length, as the alignment of the entire fragment is fixed. If the resulting fragment length falls in a *reasonable range* (by default defined as the 0.5-percentile to 99.8-percentile) then the inferred path will be accepted; otherwise we mark C is failed to bridge.

Determining false starting/ending vertices

The constructed splice graph is usually erroneous due to missing junctions, sequencing and alignment errors. We observe that erroneous starting and ending vertices (i.e., those connected to the source and sink vertices, respectively) can be identified through reads that are failed to bridge. Extended Data Figure 3 gives an example: the two blue reads and the second and the third red reads cannot be bridged, as there is no edge connecting vertices 2 and 3; these suggest that vertices 2 and 3 are false starting and ending vertices (there is likely a missing junction between vertices 2 and 3 in this example).

We design an algorithm that implements above observation. Let u and v be two consecutive vertices in the splice graph (i.e., there is no any other vertex between u and v). If u is an ending vertex (i.e., $(u,t) \in E$)

and v is a starting vertex (i.e., $(s,u) \in E$), then we will add a *pseudo-edge* (u,v) with weight of 0.5 to the splice graph. (In Extended Data Figure 3, edge $(2,3)$ will be added as pseudo-edge.) The expanded splice graph (with pseudo-edges added) will be actually used for bridging all consecutive ends by the algorithm described above. Notice that any non-pseudo-edge will have a weight at least 1; therefore, if the bottleneck-weight of the most reliable path p of a pair of ends (l,l') equals to 0.5, then we know that p must cross at least one pseudo-edge and that (l,l') cannot be bridged with non-pseudo-edges only. In this case, we will examine all pseudo-edges (u,v) in p, and assign a *pseudo-score* of value 1 to u and to v. We run this algorithm for all read groups, and the pseudo-score will be accumulated. (For example, in Extended Data Figure 3, both vertices 2 and 3 will have a pseudo-score of 2; the blue and the red reads contribute 1 each.) Intuitively, the larger pseudo-score is, the more likely the vertex is a false starting/ending vertex. Let $x(v)$ be the pseudo-score of vertex v and let $w(v)$ be the average coverage of v. We calculate $z(v) := \log(w(v)+1) - \log(x(v)+1)$ and report v is a false vertex only if $x(v) \geq 1$ and $z(v)$ is less than a threshold (by default 1.5). We do this to ensure that, starting/ending vertices with large coverage will be determined as false only if there exists a relatively significant number of supporting reads.

The determined false starting and ending vertices will be kept in the splice graph for decomposition (Step 3). A resulting *s-t* path that contains any false vertex will be classified as "transcript fragment" (as opposed to "full-length transcript"). We keep these transcript fragments as they explain the reads aligned to the false vertices (it's that we have evidence to determine they are not full-length transcripts). In our implementation full-length transcripts and transcript fragments will be written to separate files. The accuracy of assemblies reported in the Main Section are for the full-length transcripts.

Improved algorithm for phase-preserving graph decomposition

The set of bridged single-end phasing paths $H := \{h(C) \mid C \in \mathcal{C}\}$ obtained in Step 1 and the refined splice graph G' obtained in Step 2 will be used as input by the algorithm described here to construct a set of *s-t* paths P and their associated abundance $f(p)$ for any $p \in P$. Since each multi-end phasing path $C \in \mathcal{C}$ has been bridged as a single-end phasing path $h(C)$ in the splice graph G', the objective of preserving C in Problem 1 now becomes a simpler constraint of preserving $h(C)$. In our previous work Scallop[6], we designed an algorithm (hereinafter we refer to as the Scallop algorithm) that solves *exactly* the same problem of decomposing the splice graph in the presence of (single-end) phasing paths. Specifically, the

Scallop algorithm iteratively decomposes the vertices in the splice graph until it is fully decomposed: it first decomposes all unsplittable vertices, then all splittable vertices, and finally all trivial vertices. In Scallop algorithm, which unsplittable vertex is picked among all unsplittable ones is determined by calculating $\arg\min_{v:\ v \text{ is unsplittable}} z(v)$, where $z(v)$ measures the coverage deviation between the existing edges and the resulting new edges. Which splittable vertex is picked among all splittable ones is determined by calculating $\arg\min_{v:\ v \text{ is splittable}} z'(v)$, where $z'(v)$ measures the imbalancement of the resulting new vertices. All trivial vertices are picked and decomposed arbitrarily.

We observed that, how vertices in each category are prioritized for decomposition is highly related to the assembly accuracy, but the measures used in the Scallop algorithm (i.e., $z(\cdot)$ and $z'(\cdot)$, described above) may not be able to always pick the optimal vertex. To mitigate this issue, Scallop2 uses an approach that calculates and integrates multiple phase-preserving decompositions. Specifically, we decompose the splice graph N times (N is a parameter of Scallop2 with default value of 10), using the same phase-preserving algorithmic framework as in the Scallop algorithm but picking different vertices to decompose in each category. The first run in Scallop2 uses $\arg\min_{v:\ v \text{ is unsplittable}} z(v)$ to pick unsplittable vertex, and $\arg\min_{v:\ v \text{ is splittable}} z'(v)$ to pick splittable vertex, while the remaining $(N-1)$ runs pick vertex uniformly at random in each category. This will return N sets of *s*-*t* paths. We then use a consensus method to select transcripts: an *s*-*t* path p will be kept if it appears in at least $N/2$ sets, and its abundance $f(p)$ will be averaged over all its appearances. The pseudo-code that describes this algorithm is given in Supplementary Algorithm 1.

Data availability

Smart-seq3 is a single-cell protocol that generates multi-end RNA-seq data using barcoding technology. The Smart-seq3 data used in this study was published with Smart-seq3 protocol[17], publically available at https://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-8735. We use two datasets downloaded from Smart-seq3, the first dataset, referred to as HEK293T, contains 192 human cells; the second dataset, referred to as Mouse-Fibroblast, includes 369 mouse cells. For Illumina platform, we use 10 human paired-end RNA-seq samples that were downloaded from ENCODE project[22] and we refer to these 10 samples as ENCODE10. Their accession IDs are SRR307903, SRR307911, SRR315323, SRR315334, SRR387661, SRR534291, SRR534307, SRR534319, SRR545695, and SRR545723. The alignments of these samples are available at Penn State Data Commons repository[23]. Source data for Figure 1 is available with this

manuscript.

Code availability

Scallop2 is available at the Zenodo repository[24] and on GitHub (https://github.com/Shao-Group/scallop2). Scripts and documentation that reproduce the experimental results are available at the Zenodo repository[25] and on GitHub (https://github.com/Shao-Group/scallop2-test).

Statistics

All error bars represent the standard deviation.

Acknowledgments

This work is partly supported by the US National Science Foundation (DBI-2019797 to M.S.), by the US National Institutes of Health (R01HG011065 to M.S.), and by the Charles K. Etner Career Development Professorship awarded to M.S. by The Pennsylvania State University. Initial algorithmic exploration of Scallop2 advancements were conducted with Carl Kingsford and were supported by the Gordon and Betty Moore Foundation (GMBF 4554 to C.K.) and the US National Institutes of Health (R01GM122935 to C.K.). Affiliation of Carl Kingsford: Computational Biology Department, School of Computer Science, Carnegie Mellon University.

Author Contributions Statement

Q.Z., Q.S., and M.S. designed and implemented the algorithms. The experimental comparisons were primarily conducted by Q.Z. All authors drafted and approved this manuscript.

Competing Interests Statement

The authors declare no competing financial interests.

Figure Legends/Captions

Figure 1: Comparison of assembly accuracy of the four methods (Scallop2, StringTie2, Scallop, and CLASS2). (**a**) The assembly accuracy of the four assemblers on two Smart-seq3 datasets, HEK293T (HE), contains 192 human cells, and Mouse-Fibroblast (MF), contains 369 mouse cells. The mean and standard deviation are reported over all cells in each dataset. (**b**) Comparison of adjusted precision (mean and standard deviation on two Smart-seq3 datasets, HEK293T (left) and Mouse-Fibroblast (right)) between Scallop2 and each of the other three methods at the same level of sensitivity. (**c**) The assembly accuracy of the four assemblers on ENCODE10 data. The two groups of bars correspond to HISAT2 (HI) and STAR (ST) alignments, respectively. The error bars show the standard deviation over the 10 samples. (**d**) Comparison of adjusted precision (mean and standard deviation over HISAT2 (left) and STAR (right) alignments of ENCODE10) between Scallop2 and each of the other three methods at the same level of sensitivity. (**e**) Assembly accuracy on ENCODE10 dataset at individual samples. The precision-sensitivity curves are drawn for Scallop2. CLASS2 did not finish running in 11 days on sample SRR387661.

Extended Data Figure 1: Illustrating the construction of the splice graph G and the associated multi-end phasing paths C from read alignments of a gene locus. Inferred splice positions in the reference genome are marked with black bars. Exons and partial exons are labeled with numbers above the reference genome. Alignment reads with the same color represent they form a group (i.e., attached with the same barcode or being the two ends of a pair in paired-end reads); we use gray color to represent reads forming groups of their own. The read alignments are used as input to construct the splice graph and the associated multi-end phasing paths. From the given alignments 5 (partial) exons (numbered 1–5) are identified. A source vertex s and a sink vertex t are added to the splice graph. Each numbered arrow represents a directed edge and its weight.

Extended Data Figure 2: Illustration of bridging a multi-end phasing path into a single-end phasing path. Each numbered arrow represents a directed edge and its weight. Vertices and edges in the multi-end phasing path C are blue circled and arrowed respectively. Inferred bridging paths for (l_1, l_2) and (l_2, l_3) are marked red. The single-end phasing path $h(C) = (1,2,3,4,6)$ is bridged using the splice graph and multi-end phasing path $C = (l_1 = (1,2), l_2 = (4), l_3 = (6))$ as input.

Extended Data Figure 3: Illustration of identifying false starting/ending vertices. Inferred splice posi-

tions in the reference genome are marked with black bars. Exons and partial exons are labeled with numbers above the reference genome. Alignment reads with the same color represent they form a group (i.e., attached with the same barcode or being the two ends of a pair in paired-end reads); we use gray color to represent reads forming groups of their own. The read alignments are used as input to construct the splice graph and the associated multi-end phasing paths. From the given alignments 4 partial-exons (numbered 1–4) are identified. A source vertex s and a sink vertex t are added to the splice graph. Each numbered arrow represents an directed edge and its weight. The circled vertex with number 2 (respectively 3) is classified as ending (respectively starting) vertex as there is no departing (respectively entering) junction. An pseudo-edge from the circled vertex with number 2 to the circled vertex with number 3 $(2, 3)$ will be added to the splice graph for bridging. The bridging of blue reads and red reads (the second and the third ones) will cross this pseudo-edge, giving a pseudo-score of 2 for both vertices.

References

[1] C. Trapnell, B.A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M.J. Van Baren, S.L. Salzberg, B.J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat. Biotechnol.*, 28(5):511–515, 2010.

[2] M. Guttman, M. Garber, J.Z. Levin, J. Donaghey, J. Robinson, X. Adiconis, L. Fan, et al. Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. *Nat. Biotechnol.*, 28(5):503–510, 2010.

[3] A.I. Tomescu, A. Kuosmanen, R. Rizzi, and V. Mäkinen. A novel min-cost flow method for estimating transcript expression with RNA-Seq. *BMC Bioinformatics*, 14(5):1, 2013.

[4] Li Song, Sarven Sabunciyan, and Liliana Florea. CLASS2: accurate and efficient splice variant annotation from RNA-seq reads. *Nucleic Acids Res*, 44(10):e98, 2016.

[5] J. Liu, T. Yu, T. Jiang, and G. Li. TransComb: genome-guided transcriptome assembly via combing junctions in splicing graphs. *Genome Biol.*, 17(1):213, 2016.

[6] M. Shao and C. Kingsford. Accurate assembly of transcripts through phase-preserving graph decomposition. *Nature Biotechnology*, 35(12):1167–1169, 2017.

[7] M. Pertea, G.M. Pertea, C.M. Antonescu, T.-C. Chang, J.T. Mendell, and S.L. Salzberg. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nat. Biotechnol.*, 33(3): 290–295, 2015.

[8] Sam Kovaka, Aleksey V. Zimin, Geo M. Pertea, Roham Razaghi, Steven L. Salzberg, and Mihaela Pertea. Transcriptome assembly from long-read RNA-seq alignments with StringTie2. *Genome Biology*, 20:278, 2019.

[9] Shunfu Mao, Lior Pachter, David Tse, and Sreeram Kannan. RefShannon: A genome-guided transcriptome assembler using sparse flow decomposition. *PLoS One*, 15:6, 2020.

[10] L.H. Tung, M. Shao, and C. Kingsford. Quantifying the benefit offered by transcript assembly on single-molecule long reads. *Genome Biology*, 20(1):1–18, 2019.

[11] M. Shao and C. Kingsford. Theory and a heuristic for the minimum path flow decomposition problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16:658–670, 2017.

[12] Lucia Williams, Alexandru I. Tomescu, and Brendan Mumey. Flow Decomposition with Subpath Constraints. In *21st International Workshop on Algorithms in Bioinformatics (WABI)*, volume 201, pages 16:1–16:15, 2021.

[13] Lucia Williams, Gill Reynolds, and Brendan Mumey. RNA transcript assembly using inexact flow. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, page 1907–1914, 2019.

[14] Shanika L Amarasinghe, Shian Su, Xueyi Dong, Luke Zappia, Matthew E. Ritchie, and Quentin Gouil. Opportunities and challenges in long-read sequencing data analysis. *Genome Biology*, 21:30, 2020.

[15] Adam Voshall and Etsuko N. Moriyama. Next-generation transcriptome assembly: strategies and performance analysis. *Bioinformatics in the Era of Post Genomics and Big Data*, pages 15–36, 2018.

[16] Simone Picelli, Omid R Faridani, Åsa K Björklund, Gösta Winberg, Sven Sagasser, and Rickard Sandberg. Full-length rna-seq from single cells using Smart-seq2. *Nature Protocols*, 9:171–181, 2014.

[17] Michael Hagemann-Jensen, Christoph Ziegenhain, Ping Chen, Daniel Ramsköld, Gert-Jan Hendriks,

Anton J. M. Larsson, Omid R. Faridani, and Rickard Sandberg. Single-cell RNA counting at allele and isoform resolution using Smart-seq3. *Nature Biotechnology*, 38:708–714, 2020.

[18] Geo Pertea and Mihaela Pertea. GFF utilities: GffRead and GffCompare. *F1000 Research*, 9, 2020.

[19] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley and Steven L Salzberg. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, 14:R36, 2013.

[20] A. Dobin, C.A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T.R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

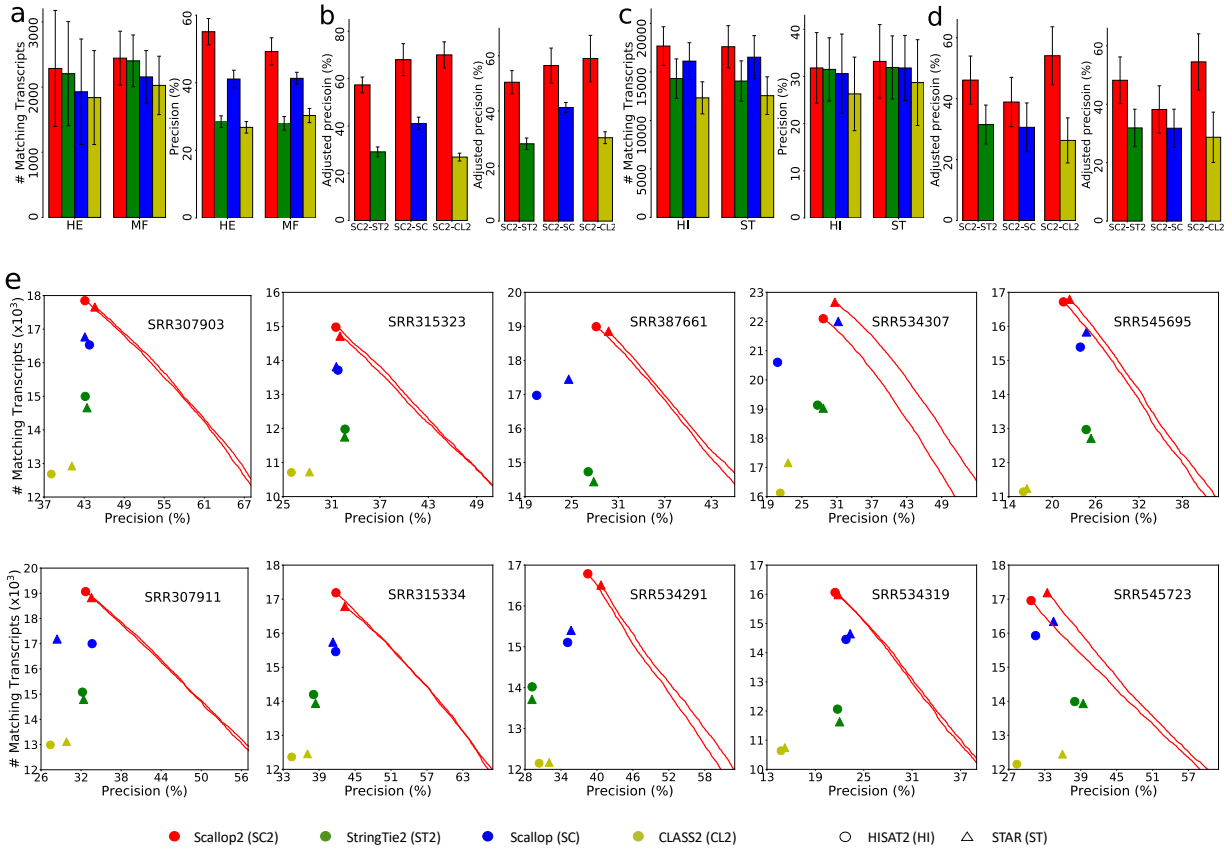
[21] D. Kim, B. Langmead, and S.L. Salzberg. HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods*, 12(4):357–360, 2015.

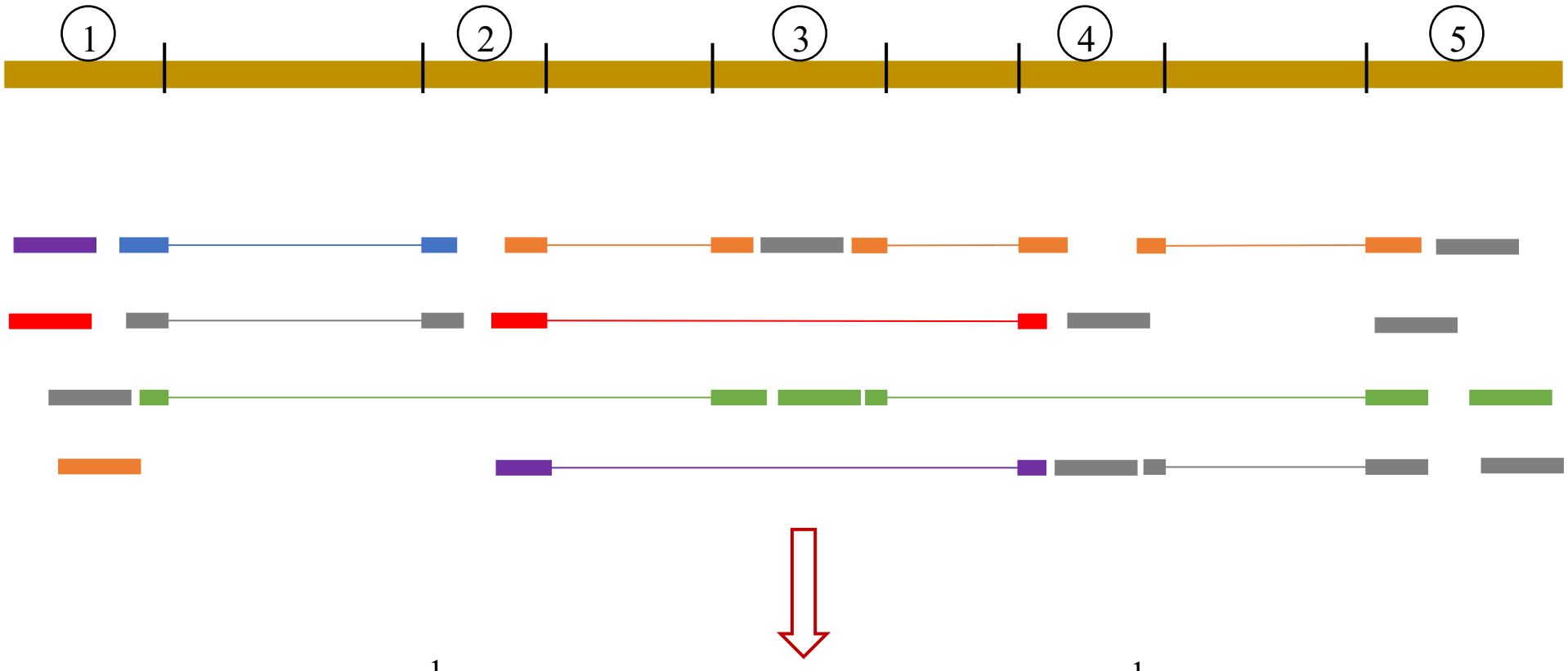
[22] ENCODE Project Consortium et al. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, 2012.

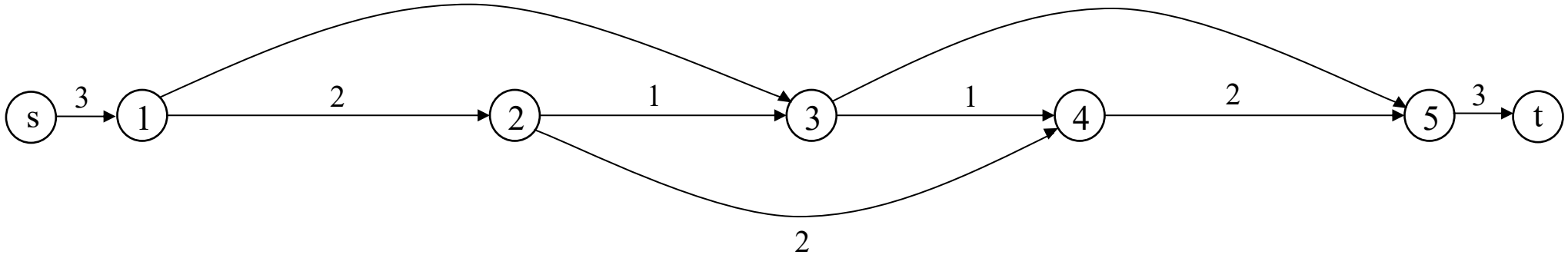
[23] Q. Shi and M. Shao. ENCODE10 dataset. 2020. doi: doi:10.26208/8c06-w247. Penn State Data Commons.

[24] Q Zhang, Q. Shi, and M. Shao. Code for scallop2. 2022. doi: doi.org/10.5281/zenodo.6013717. Zenodo.

[25] Q Zhang, Q. Shi, and M. Shao. Code for scallop2-test. 2022. doi: doi.org/10.5281/zenodo.6064927. Zenodo.


a
b
c
d
e
Matching Transcripts
Precision (%)
Adjusted precisoin (%)
HE
MF
HI
ST
SC2-ST2
SC2-SC
SC2-CL2
SRR307903
SRR315323
SRR387661
SRR534307
SRR545695
SRR307911
SRR315334
SRR534291
SRR534319
SRR545723
Matching Transcripts (x10³)
Scallop2 (SC2)
StringTie2 (ST2)
Scallop (SC)
CLASS2 (CL2)
HISAT2 (HI)
STAR (ST)





+

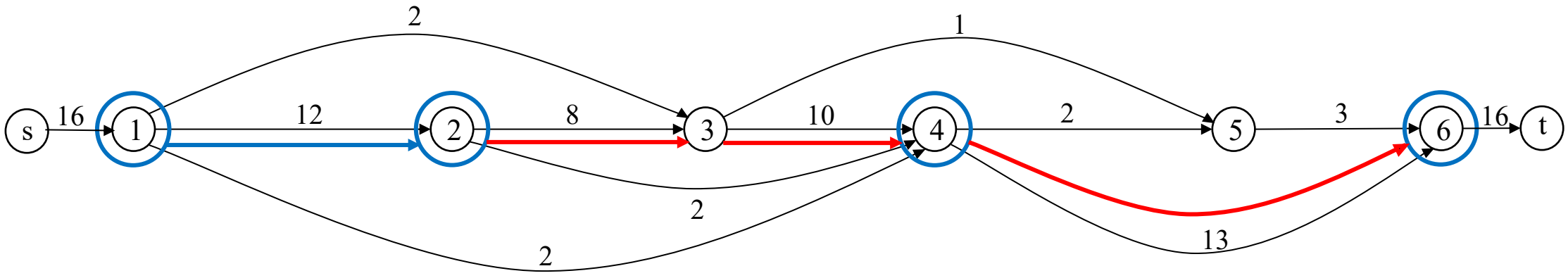
$((1,2))$

$((1),(2,4)) = ((1),(2,4))$

$((1,3),(3),(3,5),(5))$

$((1),(2,3),(3,4),(4,5))$

multi-end phasing paths



splice graph

$((1, 2), (4), (6))$

multi-end phasing path

$((1, 2), (4)) \rightarrow (1, 2, 3, 4)$

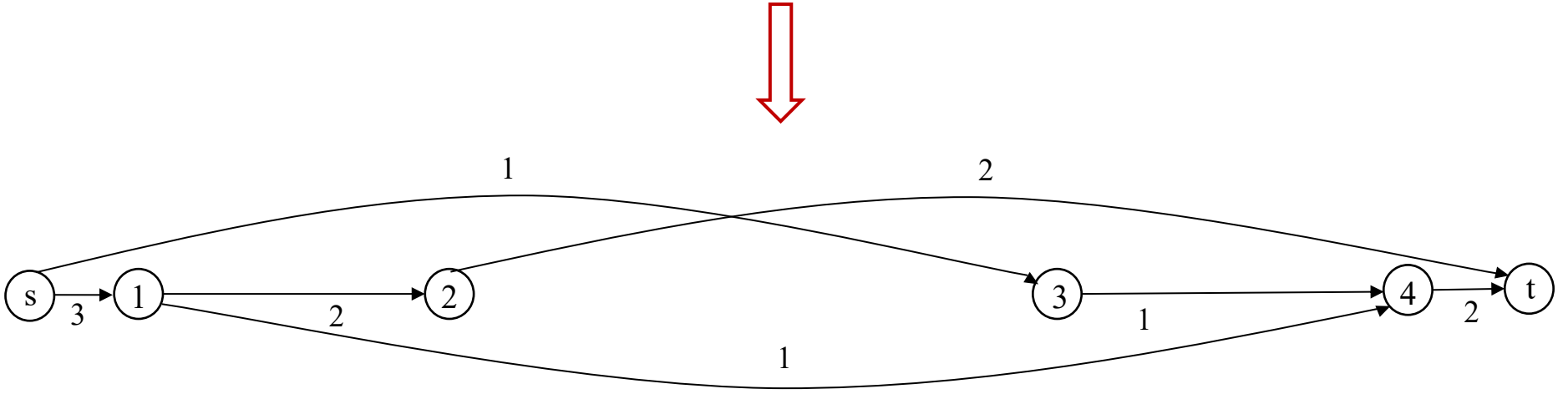
$((4), (6)) \rightarrow (4, 6)$

$\rightarrow (1, 2, 3, 4, 6)$

single-end phasing path

reference genome

read alignments



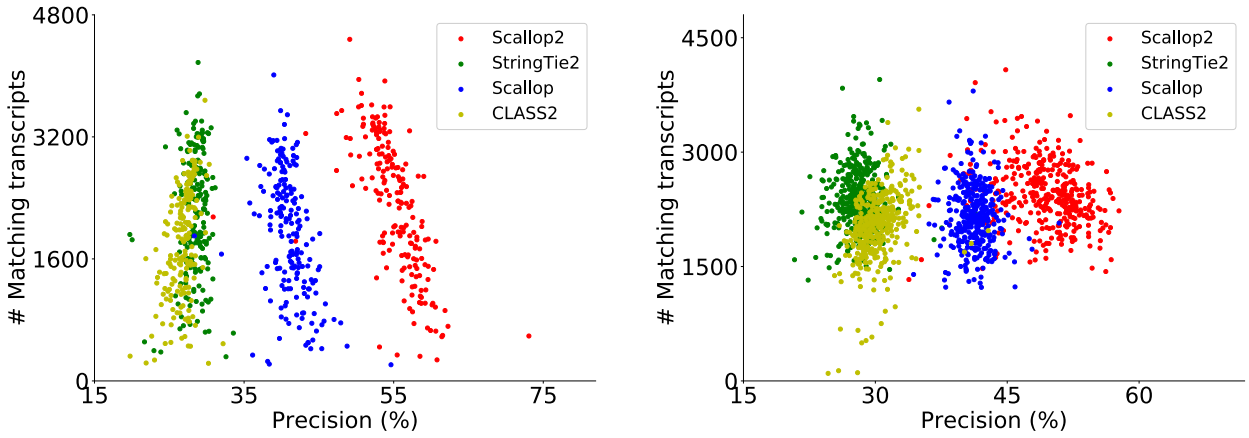
splice graph

((1), (3))

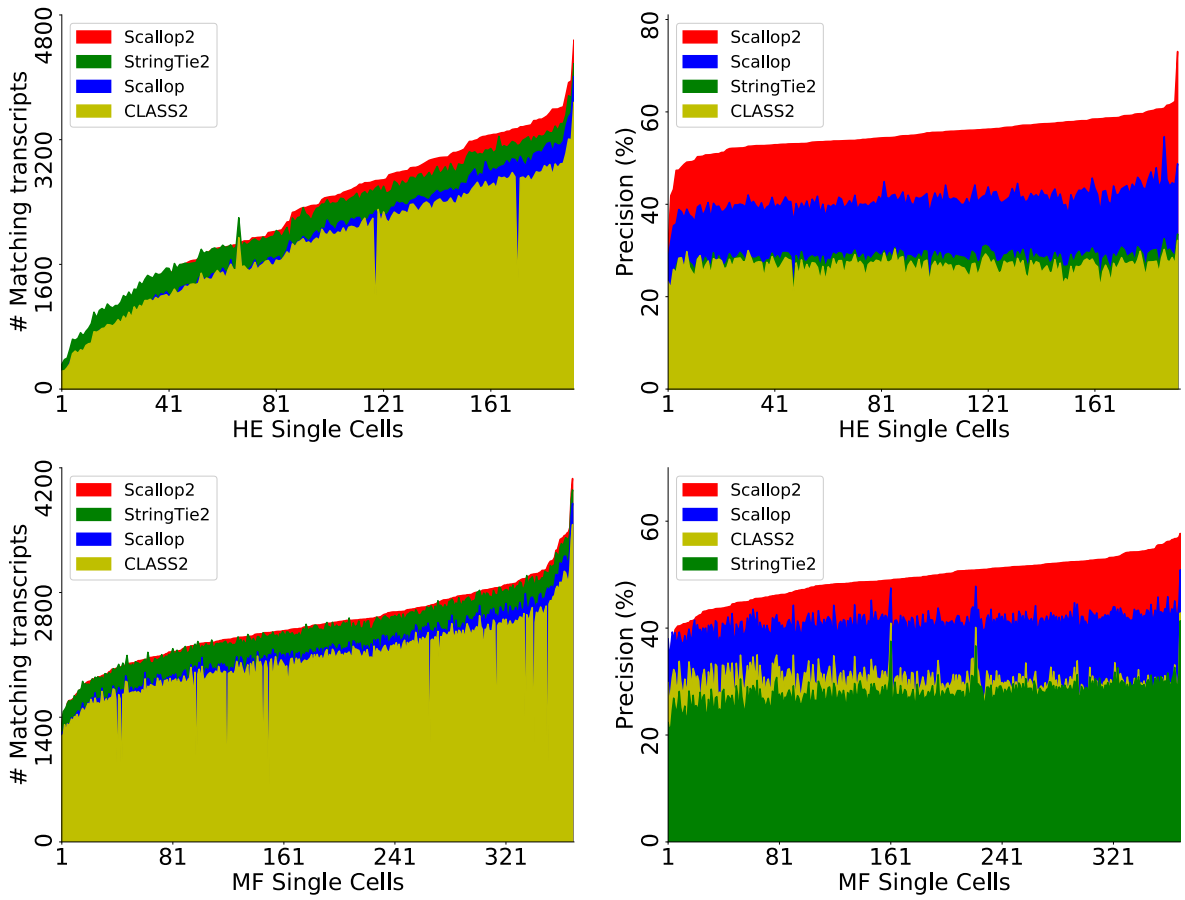
((1), (2), (4))

multi-end phasing paths

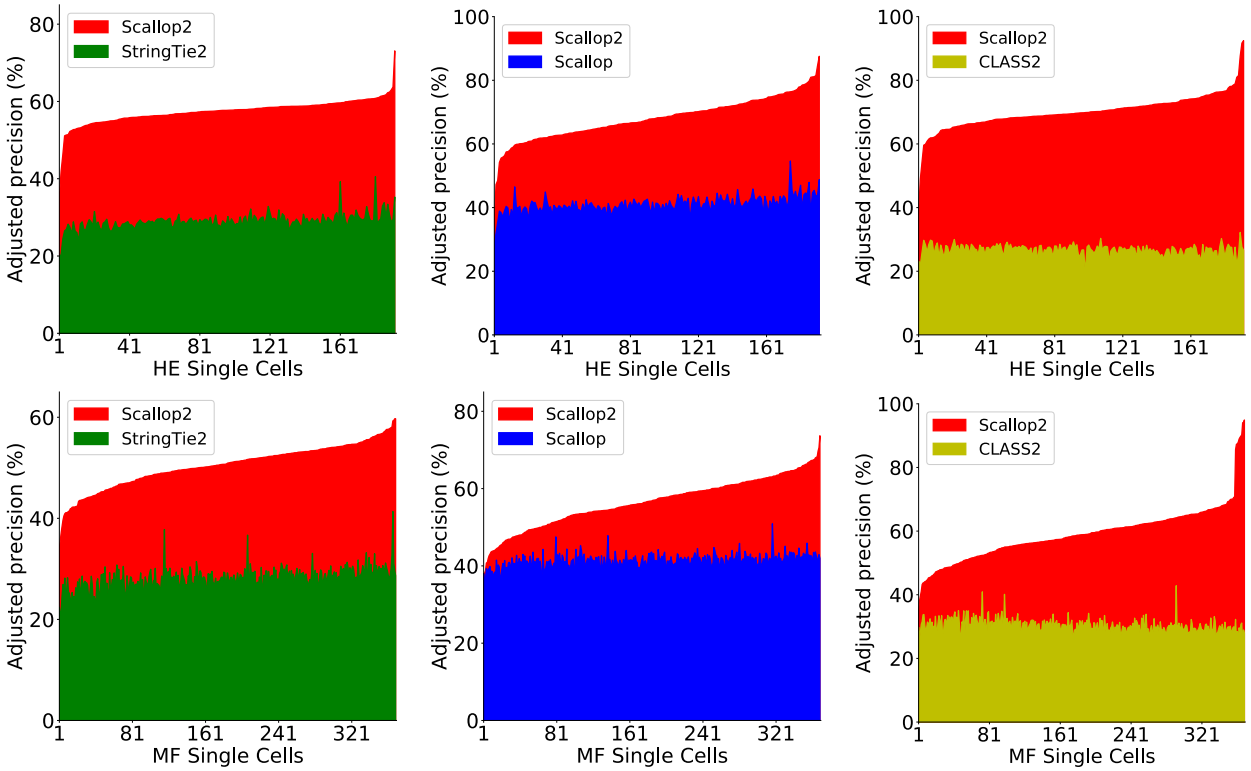
Supplementary Figures



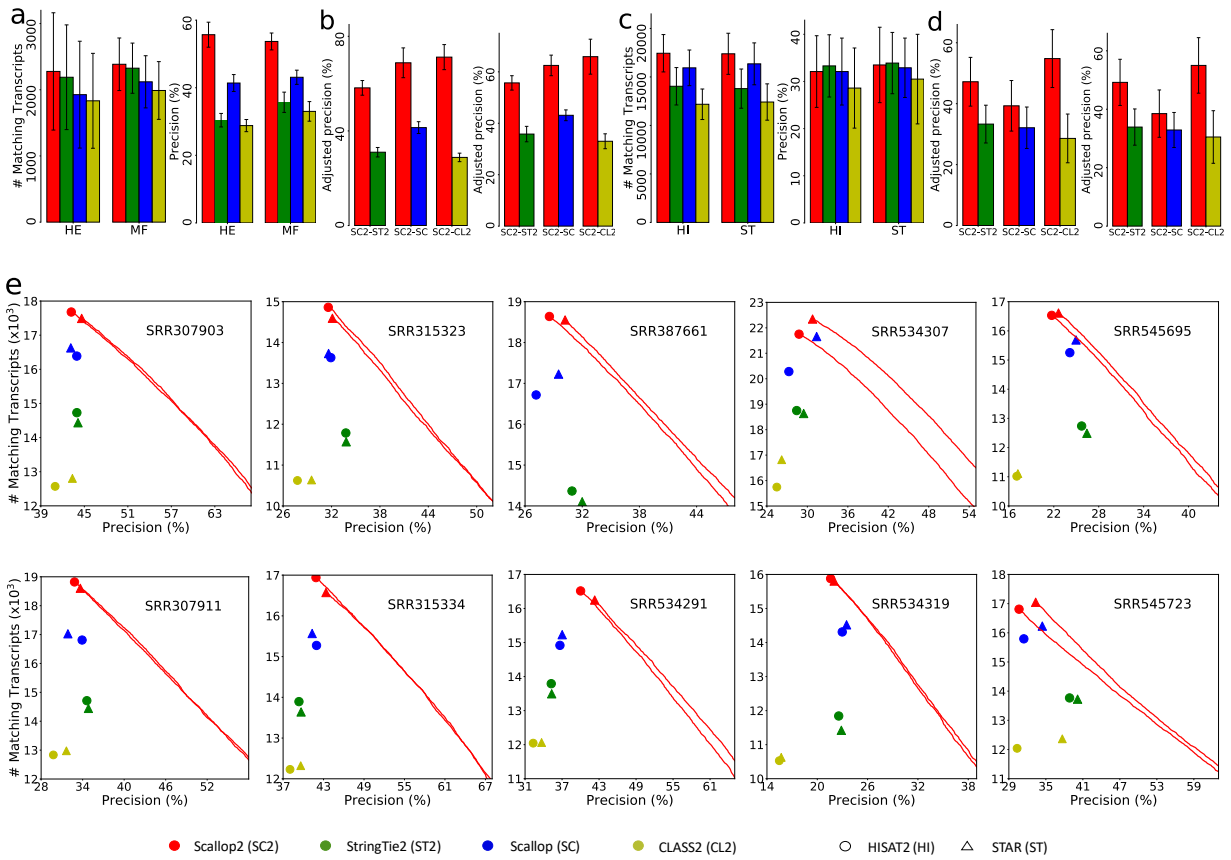
Supplementary Figure 1: Assembly accuracy at individual cells of HEK293T (left panel) and Mouse-Fibroblast (right panel).



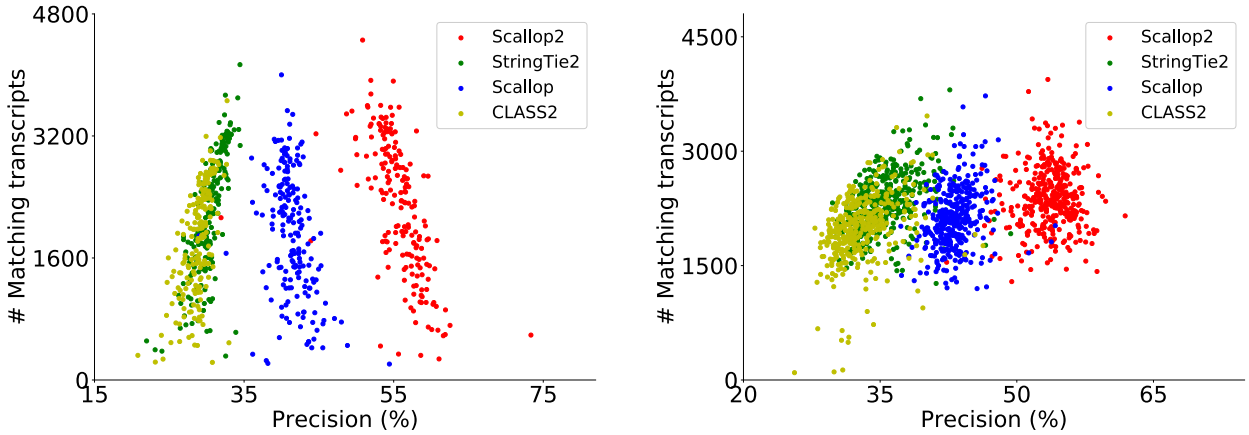
Supplementary Figure 2: Assembly accuracy at individual cells on two Smart-seq3 datasets, HEK293T (HE) and Mouse-Fibroblast (MF). In each panel, cells are sorted in ascending order w.r.t. the accuracy of Scallop2 (so the curves for Scallop2 are monotonic).



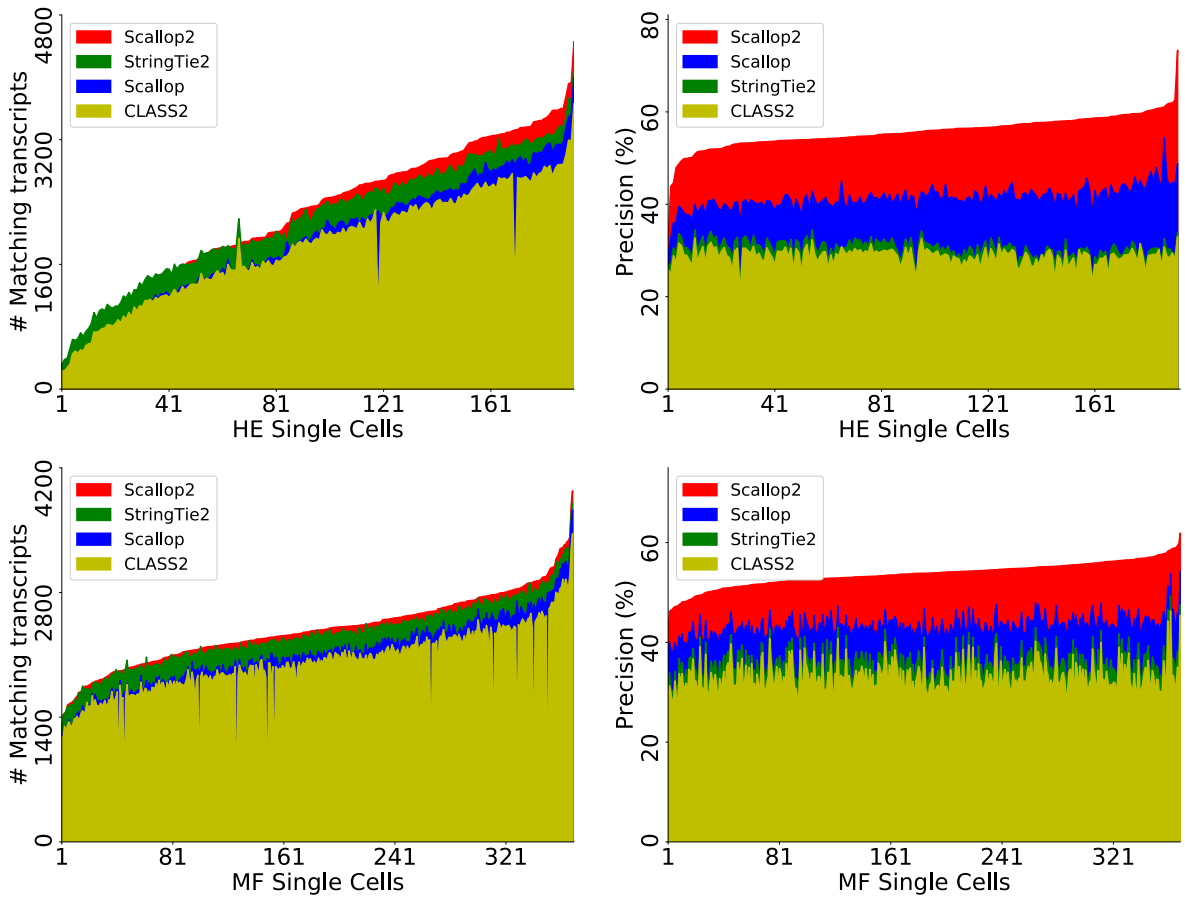
Supplementary Figure 3: Comparison of adjusted precision between Scallop2 and each of the other three methods at the same level of sensitivity on two Smart-seq3 datasets, HEK293T (HE) and Mouse-Fibroblast (MF). In each panel, cells are sorted in ascending order w.r.t. the accuracy of Scallop2 (so the curves for Scallop2 are monotonic).



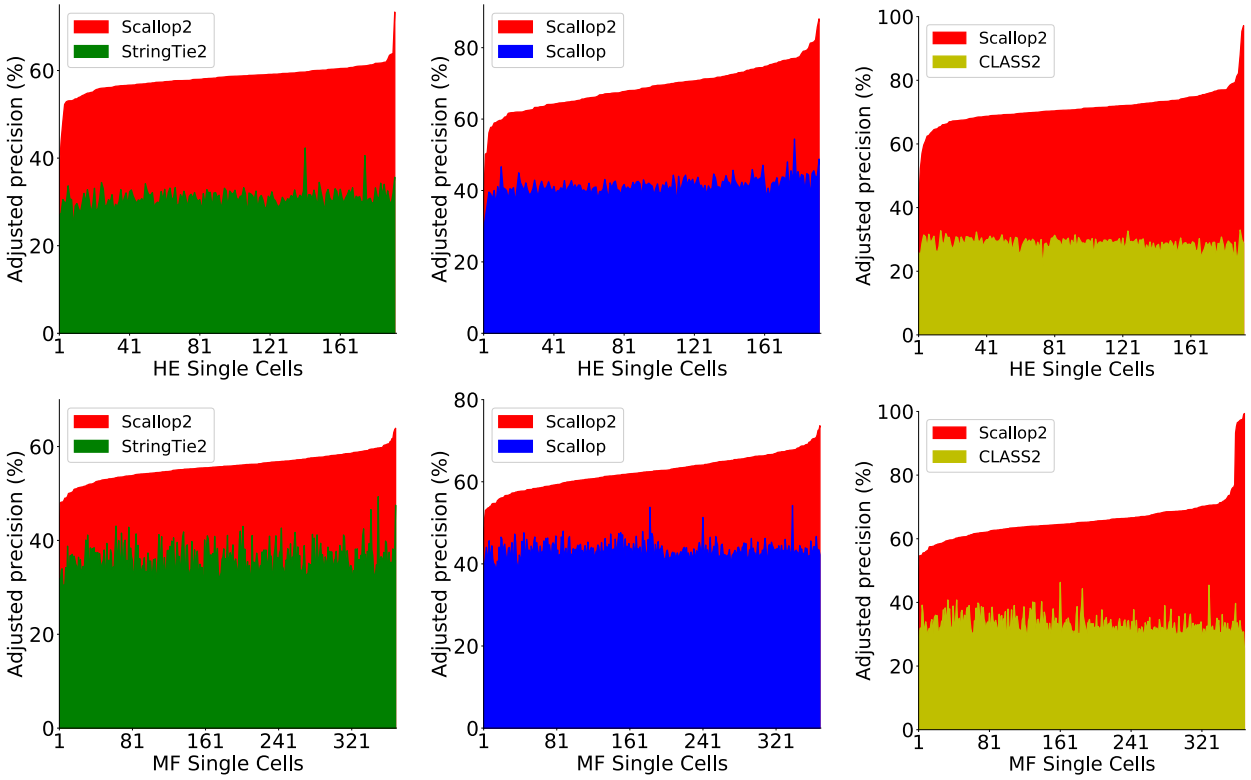
Supplementary Figure 4: Comparison of assembly accuracy for the multi-exon transcripts assembled by the four methods (Scallop2, StringTie2, Scallop, and CLASS2). (**a**) The assembly accuracy for multi-exon transcripts on two Smart-seq3 datasets, HEK293T (HE) and Mouse-Fibroblast (MF). The mean and standard deviation are reported over all cells in each dataset. (**b**) Comparison of adjusted precision (mean and standard deviation on two Smart-seq3 datasets, HEK293T (left) and Mouse-Fibroblast (right)) between Scallop2 and each of the other three methods at the same level of sensitivity. (**c**) The assembly accuracy for multi-exon transcripts on 10 paired-end RNA-seq samples (named ENCODE10). The two groups of bars correspond to HISAT2 (HI) and STAR (ST) alignments, respectively. The error bars show the standard deviation over the 10 samples. (**d**) Comparison of adjusted precision (mean and standard deviation over HISAT2 (left) and STAR (right) alignments of ENCODE10) between Scallop2 and each of the other three methods at the same level of sensitivity. (**e**) Assembly accuracy for multi-exon transcripts on ENCODE10 at individual samples. The precision-sensitivity curves are drawn for Scallop2.



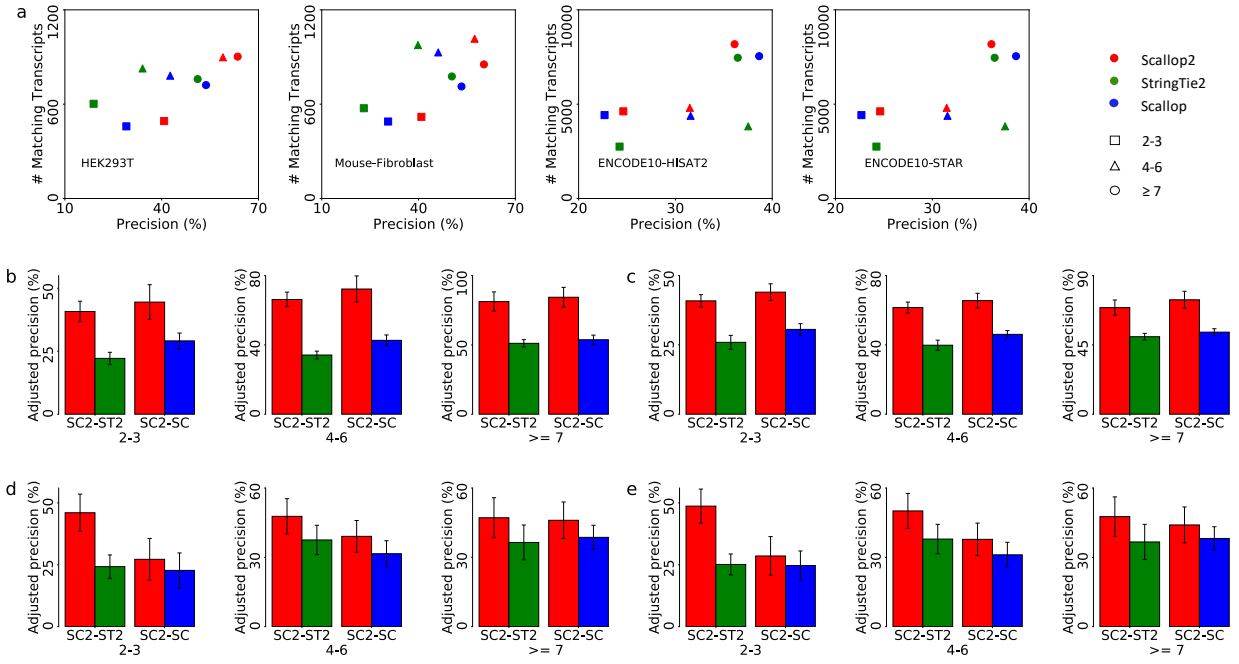
Supplementary Figure 5: Assembly accuracy for multi-exon transcripts at individual cells of HEK293T (left panel) and Mouse-Fibroblast (right panel).



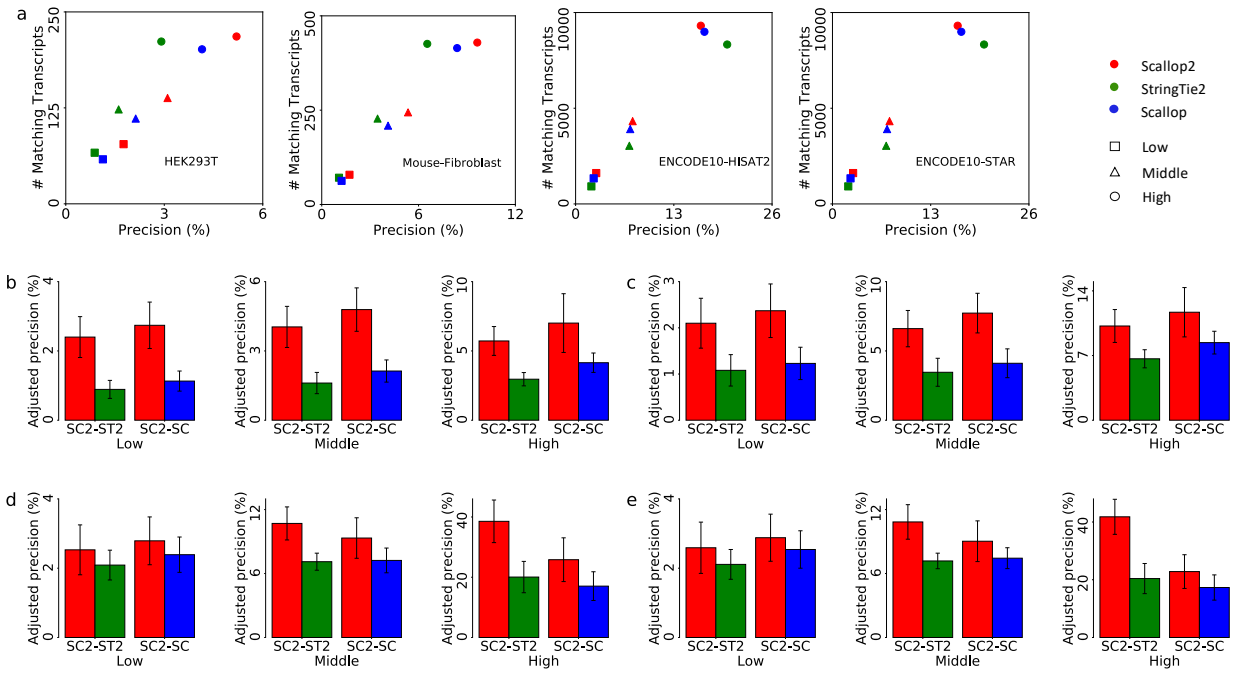
Supplementary Figure 6: Assembly accuracy for multi-exon transcripts at individual cells on two Smart-seq3 datasets, HEK293T (HE) and Mouse-Fibroblast (MF). In each panel, cells are sorted in ascending order w.r.t. the accuracy of Scallop2 (so the curves for Scallop2 are monotonic).



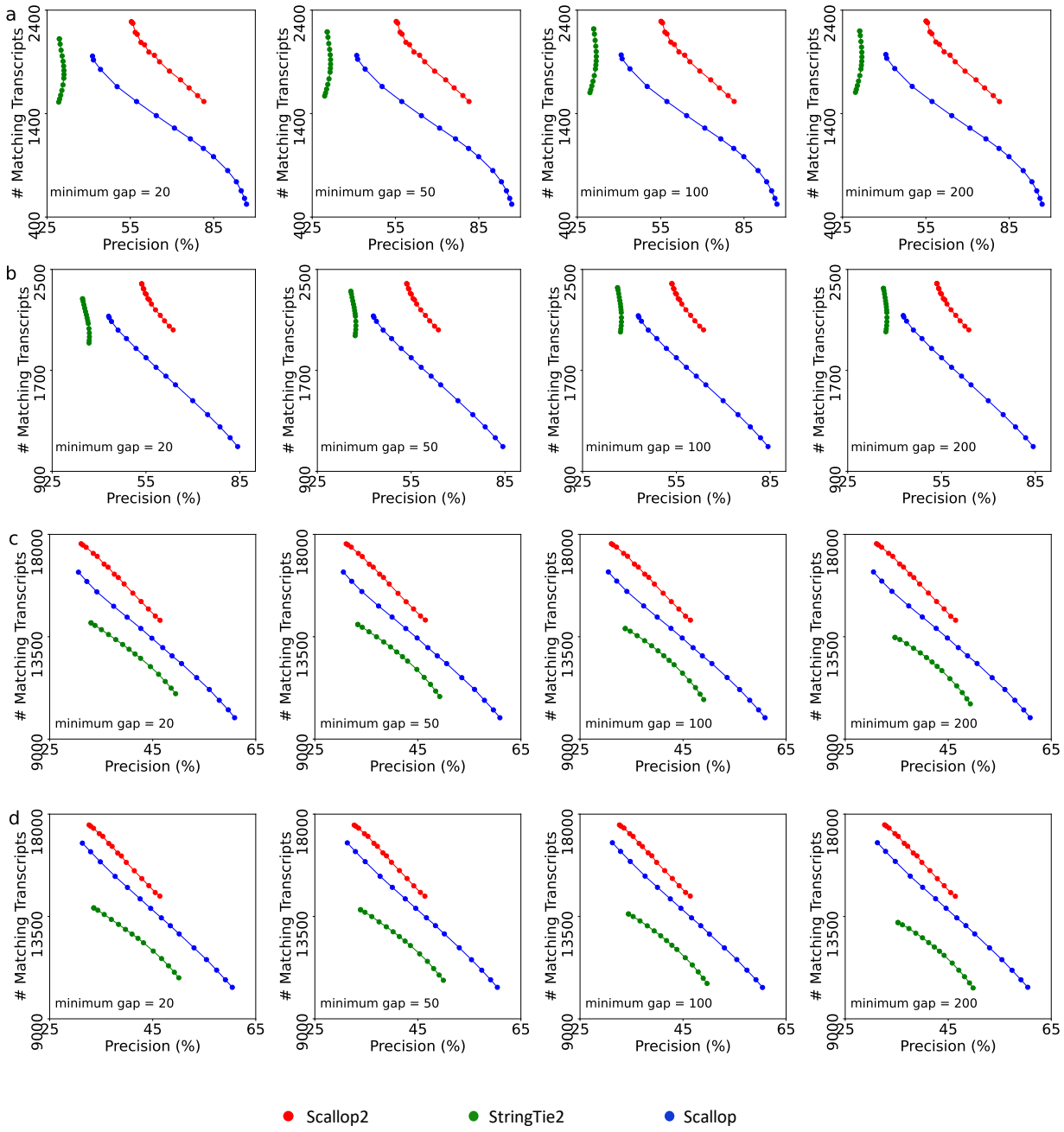
Supplementary Figure 7: Comparison of adjusted precision for multi-exon transcripts between Scallop2 and each of the other three methods at the same level of sensitivity on two Smart-seq3 datasets, HEK293T (HE) and Mouse-Fibroblast (MF). In each panel, cells are sorted in ascending order w.r.t. the accuracy of Scallop2 (so the curves for Scallop2 are monotonic).



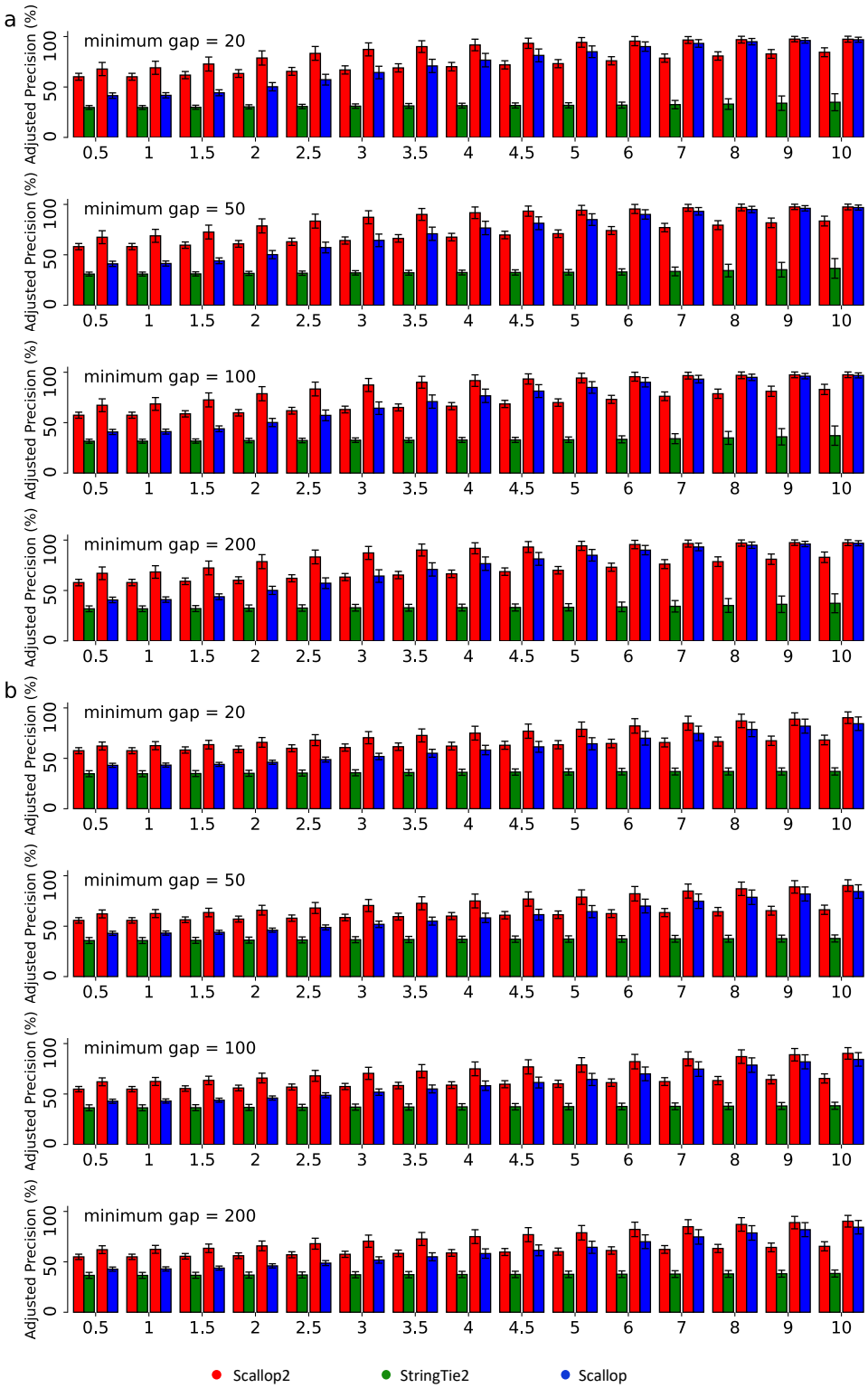
Supplementary Figure 8: Comparison of assembly accuracy for transcripts with different number of exons. Predicted transcripts and ground-truth transcripts are all partitioned into 3 categories, namely transcripts with 2–3, 4–6, and at least 7 exons, respectively. The accuracy of each category is then evaluated separately. All assemblers are running with minimum coverage set to 0.001. (**a**) Comparison of the number of matching transcripts and precision (left to right: HEK293T dataset, Mouse-Fibroblast dataset, ENCODE10 aligned with HISAT2, ENCODE10 aligned with STAR). (**b**) Comparison of adjusted precision on Smart-seq3 HEK293T dataset. (**c**) Comparison of adjusted precision on Smart-seq3 Mouse-Fibroblast dataset. (**d**) Comparison of adjusted precision on HISAT2 alignments on ENCOEDE10 dataset. (**e**) Comparison of adjusted precision on STAR alignments on ENCOEDE10 dataset.



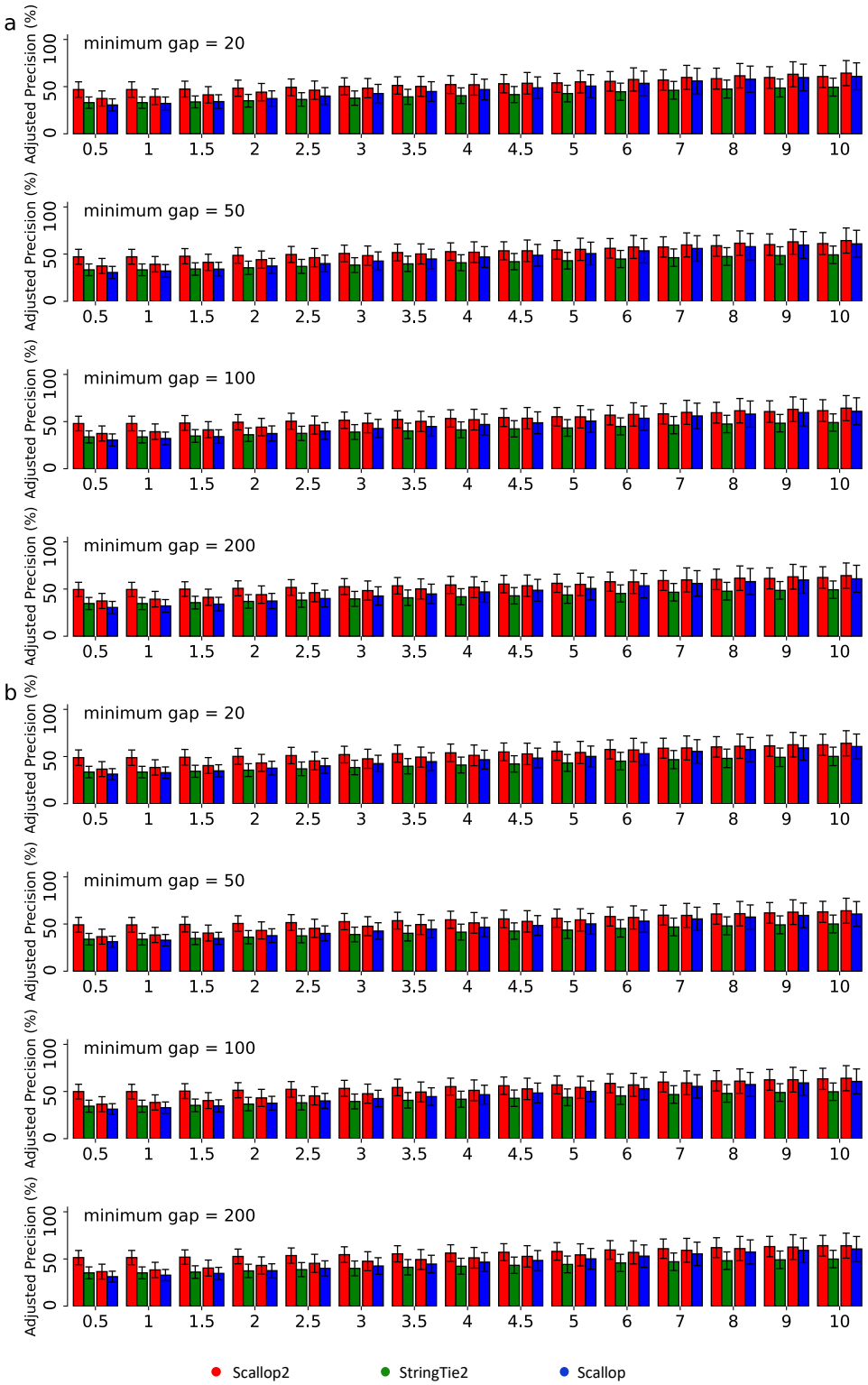
Supplementary Figure 9: Comparison of assembly accuracy for transcripts at different expression levels. The ground-truth transcripts are partitioned into 3 categories, namely low, middle, and high levels, and the predicted sets of transcripts are then evaluated using each category. All assemblers are running with minimum coverage set to 0.001. (**a**) Comparison of the number of matching transcripts and precision (left to right: HEK293T dataset, Mouse-Fibroblast dataset, ENCODE10 aligned with HISAT2, ENCODE10 aligned with STAR). (**b**) Comparison of adjusted precision on Smart-seq3 HEK293T dataset. (**c**) Comparison of adjusted precision on Smart-seq3 Mouse-Fibroblast dataset. (**d**) Comparison of adjusted precision on HISAT2 alignments on ENCOEDE10 dataset. (**e**) Comparison of adjusted precision on STAR alignments on ENCOEDE10 dataset.



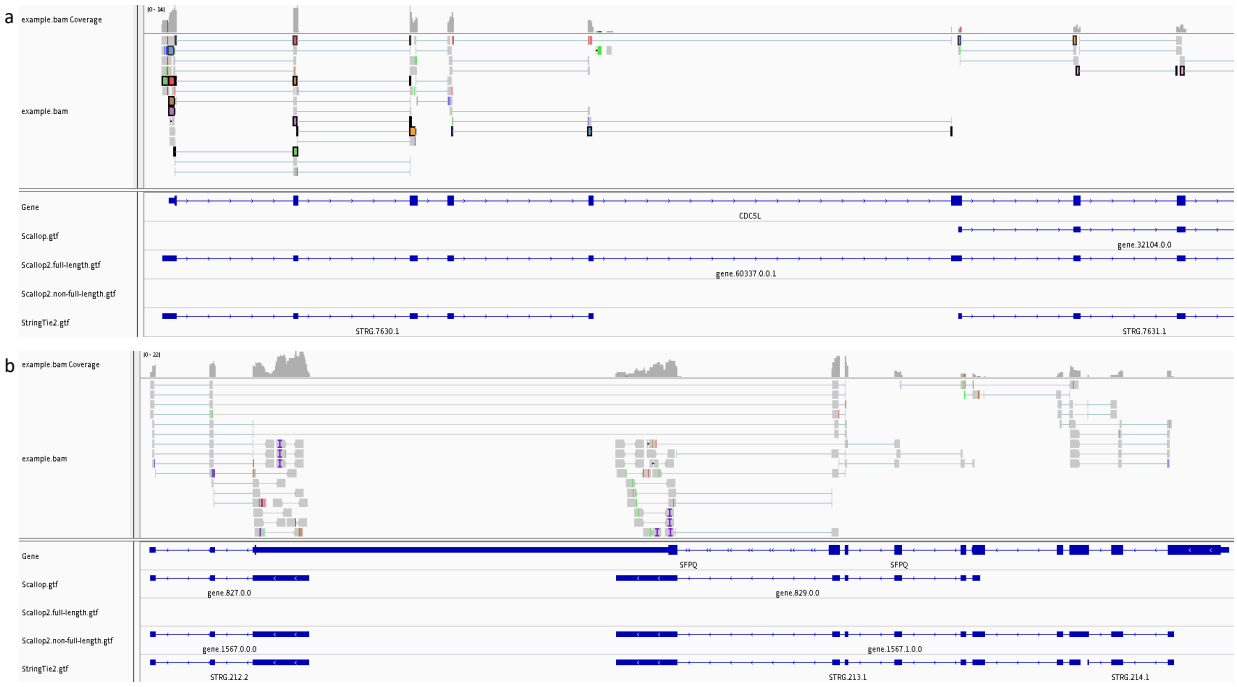
Supplementary Figure 10: Comparison of assembly accuracy on multi-exon transcripts of the three methods (StringTie2, Scallop, and Scallop2) with different settings of parameters. We vary two parameters, minimum coverage and minimum gap, and report the average accuracy over all cells or samples in each dataset. The results for each dataset consists of four subfigures (from left to right), corresponding to 4 different minimum gap thresholds {20, 50, 100, 200}. Each curve connects 15 points, corresponding to 15 different minimum coverage thresholds {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8, 9, 10}. (**a**) Assembly accuracy on Smart-seq3 HEK293T dataset. (**b**) Assembly accuracy on Smart-seq3 Mouse-Fibroblast dataset. (**c**) Assembly accuracy on HISAT2 alignments of ENCODE10 dataset. (**d**) Assembly accuracy on STAR alignments of ENCODE10 dataset.



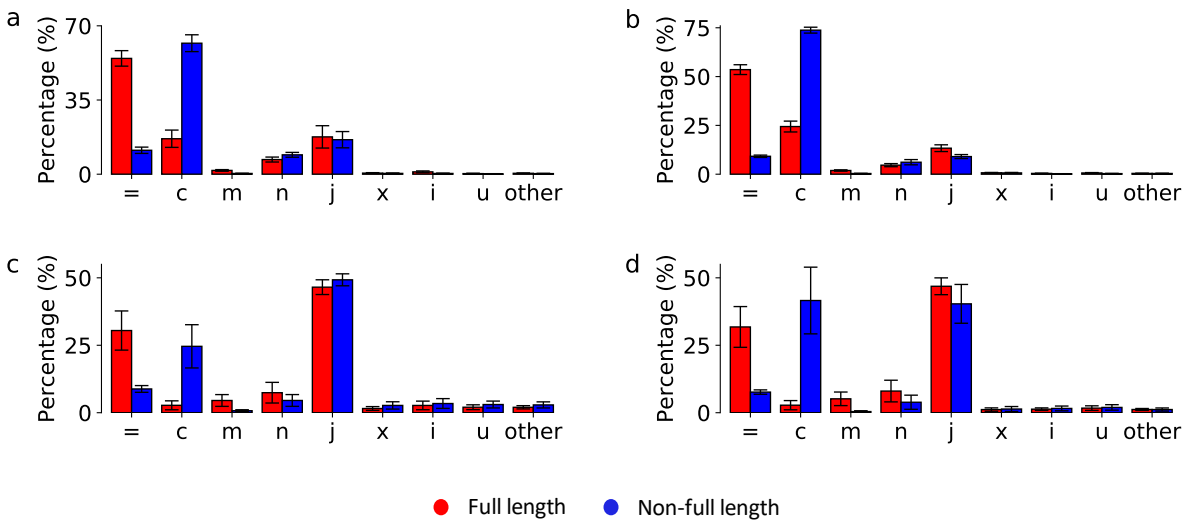
Supplementary Figure 11: Comparison of adjusted precision on multi-exon transcripts with varying parameters (the same with Supplementary Figure 10). The mean and standard deviation of adjusted precision are showed over all cells/samples in each dataset. The results of each dataset consists of four subfigures, corresponding to the 4 different minimum gap thresholds {20, 50, 100, 200}. Each subfigure consists of 15 groups, corresponding to the 15 different minimum coverage thresholds {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8, 9, 10}. (**a**) The comparison of adjusted precision on Smart-seq3 HEK293T dataset. (**b**) The comparison of adjusted precision on Smart-seq3 Mouse-Fibroblast dataset.



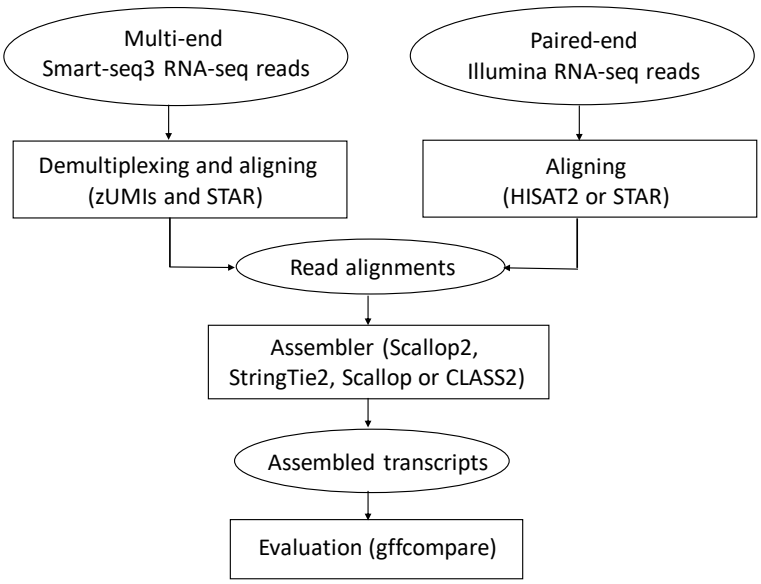
Supplementary Figure 12: Comparison of adjusted precision on multi-exon transcripts with varying parameters (the same with Supplementary Figure 10). The mean and standard deviation of adjusted precision are showed over all cells/samples in each dataset. The results of each dataset consists of four subfigures, corresponding to the 4 different minimum gap thresholds {20, 50, 100, 200}. Each subfigure consists of 15 groups, corresponding to the 15 different minimum coverage thresholds {0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8, 9, 10}. (**a**) The comparison of adjusted precision on HISAT2 alignments of ENCODE10 dataset. (**b**) The comparison of adjusted precision on STAR alignments of ENCODE10 dataset.



Supplementary Figure 13: Two gene loci visualized using IGV showing the alignment, coverage, junctions, and assembled transcripts by StringTie2, Scallop, Scallop2 (both full-length and non-full-length ones). (**a**) Reads highlighted in the same color are from the same read group. The bridging algorithm used in Scallop2 helps to fill the gap in the 5th and 6th exons which lead to the correct assembly. (**b**) Scallop2 identifies the two middle exons (with highest coverage) as false boundary vertices and therefore mark them as non-full length transcripts.



Supplementary Figure 14: Distribution of the classifications (by gffcompare) of the predicted full-length and non-full-length transcripts by Scallop2 (run with default parameters). The mean and standard deviation are reported over all cells/samples in each dataset. The results of each dataset consists of 9 groups, corresponding to the 9 major classification codes {=, c, m, n, j, x, i, u, and other}. (**a**) The comparison on Smart-seq3 HEK293T dataset; the averaged number of full-length and non-full-length transcripts are 4167 and 3312, respectively. (**b**) The comparison on Smart-seq3 Mouse-Fibroblast dataset; the averaged number of full-length and non-full-length transcripts are 4451 and 2889, respectively. (**c**) The comparison on HISAT2 alignments of ENCODE10 dataset; the averaged number of full-length and non-full-length transcripts are 57256 and 15788, respectively. (**d**) The comparison on STAR alignments of ENCODE10 dataset; the averaged number of full-length and non-full-length transcripts are 54695 and 22804, respectively.



Supplementary Figure 15: Pipeline of evaluating the performance of four compared assemblers, StringTie2, Scallop, CLASS2, and Scallop2.

Supplementary Tables

Supplementary Table 1: Comparison of CPU time and peak memory of the 4 methods on Smart-seq3 data. For each dataset, mean and standard deviation over all cells are reported.

	HEK293T		Mouse-Fibroblast	
assembler	running time (s)	memory (MB)	running time (s)	memory (MB)
Scallop2	43.1±21.3	49.1±25.9	49.4±15.3	73.2±30.7
StringTie2	8.0±3.7	15.4±5.0	10.3±2.8	29.5±10.3
Scallop	7.7±4.5	45.8±25.3	10.8±4.6	63.2±27.9
CLASS2	141.2±51.3	545.8±19.4	102.3±22.9	543.7±8.6

Supplementary Table 2: Comparison of CPU time and peak memory of the 4 methods on the ENCODE10 dataset. For each aligner used, mean and standard deviation over the 10 samples are reported.

	HISAT2 alignments		STAR alignments	
assembler	running time (s)	memory (MB)	running time (s)	memory (MB)
Scallop2	3665±3808	6123±5069	3135±3184	6041±4773
StringTie2	482±195	148±64	475±208	250±112
Scallop	949±937	8475±9426	776±545	7839±7960
CLASS2	360869±433733	2429±1108	8596±7305	2045±478

Supplementary Notes

Supplementary Note 1: Methods compared. We compare Scallop2 (version 1.1.2) with three other recent reference-based transcript assemblers, StringTie2 (version 2.1.7), Scallop (version 0.10.5), and CLASS2 (version 2.1.7). All methods were run with their default parameters, except that on the Smart-seq3 data Scallop were run with `--min_num_hits_in_bundle 5` which allows Scallop to assemble gene loci with 5 or more reads aligned (Scallop2 also has this parameter and its default value is 5; the default value of this parameter used by Scallop is 20). This is because Smart-seq3 admits lower coverage and it is favorable to use a small value to capture more transcripts.

Supplementary Note 2: Pipeline and evaluation. We use the pipeline illustrated in Supplementary Figure 15 to evaluate the performance of all compared assemblers. For Smart-seq3 data, the raw sequencing data will be demultiplexed and preprocessed using zUMIs tool, in which STAR will be called to generate reads alignments. The 10 paired-end Illumina RNA-seq samples are aligned with two popular aligners, STAR[1] and HISAT2[2]. The read alignments of each individual cell or sample will be piped to the compared assemblers.

The accuracy of the assembled transcripts will be evaluated using gffcompare[3]. We use the annotated transcriptomes as reference (Ensembl GRCh38.104 for human data and Ensembl GRCm39.104 for mouse data). We use the "transcript level" metrics defined by gffcompare: an assembled multi-exon transcript is considered as "matching" if its intron-chain exactly matches that of a transcript in the reference; an assembled single-exon transcript is defined as "matching" if there is a significant overlap (80% by default) with a single-exon transcript in the reference. We use two metrics calculated by gffcompare: the total number of matching transcripts, which is proportional to sensitivity, and precision, defined as the ratio between the total number of matching transcripts and the total number of assembled (predicted) transcripts. The multi-exon accuracy reported in Supplementary Figures 4–7 only consider multi-exon transcripts.

We note that above metrics measured w.r.t. the current transcriptome underestimates the accuracy, as novel transcripts that are correctly assembled will be considered as "not-matching" because they do not yet exist in the reference. Nevertheless, we believe such metrics are fair for the compared assemblers, as they reflect their *relative* accuracy, and yet we do not know the true expressed transcripts in these biological samples.

Supplementary Note 3: Comparison at the same level of sensitivity. We note that the idea of comparing different methods at the same level of sensitivity was first used in Scallop[4] and the approach of computing adjusted precision was also described in the Scallop paper. Here we reiterate them to make the content self-inclusive. Notice that one method might give higher number of matching transcripts but lower precision, or vise versa, on certain samples. To enable comparing in these cases, we first balance them in sensitivity and then compare them using a measure called *adjusted precision*. Specifically, let k_1 and k_2 be the number of matching transcripts obtained by two methods X_1 and X_2, and p_1 and p_2 be their corresponding precision. Assume that $k_1 > k_2$. We sort the assembled transcripts obtained by X_1 based on the expression abundance, and gradually filter out assembled transcripts with lowest abundance. In this way, the matching transcripts of X_1 will decrease while its precision will likely increase (as lowly-expressed ones are more likely to be false positive transcripts). We calculate the corresponding new measures of X_1, denoted as k_1' and p_1', as filtering goes, and stop when $k_1' = k_2$. We then report p_1' at this time as the *adjusted precision* of X_1 (now the number of matching transcripts of X_1 becomes $k_1' = k_2$). Note that the relationship between p_1', and p_2' can therefore reflect the overall performance of the two methods, as they are comparing at the same level of sensitivity.

The precision-sensitivity curves of Scallop2 illustrated in **Figure 1e** and Supplementary Figure 4e are also obtained by gradually filtering out the assembled transcripts with the lowest abundance. Therefore, the adjusted precision of Scallop2 w.r.t. another method is the x coordinate of the point on the curve whose y coordinate is equal to the point corresponding to the compared method.

Supplementary Note 4: Comparison on assembling transcripts with different number of exons. We evaluate the accuracies of the three methods (StringTie2, Scallop, and Scallop2) on assembling transcripts with different number of exons. We divide the reference transcriptome (for human it is Ensembl GRCh38.104 and for mouse it is Ensembl GRCm39.104) into three disjoint subsets corresponding to transcripts with 2-3 exons, 4-6 exons, and 7 or more exons. We use the above subsets as ground-truth transcripts. We then divide the predicted transcripts assembled by each method into three subsets with the same criterion. We then use gffcompare to compute the assembly accuracy of each predicted subset using its corresponding ground truth subset, and report the number of matching transcripts (proportional to sensitivity) and precision.

The results are illustrated in Supplementary Figure 8. Scallop2 identifies more matching transcripts with 4 or more exons, especially on Smart-seq3 single-cell datasets. Averaged over 561 Smart-seq3 single cells, Scallop2 identifies 12.9% and 21.5% more matching transcripts than StringTie2 and Scallop in the set of transcripts with 7 or more exons; 5.7% and 11.3% more matching transcripts than StringTie2 and Scallop in the set of transcripts with 4-6 number of exons. Compared at the same level of sensitivity, Scallop2 keeps the highest adjusted precision on all groups on all datasets. Averaged over 561 cells in two Smart-seq3 datasets, Scallop2 improves 84.0% and 52.9% in adjusted precision compared with StringTie2 and Scallop on transcripts with 2-3 exons; Scallop2 improves 93.8% and 69.5% in adjusted precision compared with StringTie2 and Scallop on transcripts with 4-6 exons; Scallop2 improves 58.9% and 57.0% in adjusted precision compared with StringTie2 and Scallop with 7 or more exons. Averaged over 10 Illumina RNA-seq samples, in adjusted precision Scallop2 improves 89.9% and 19.7%, 27.3% and 23.8%, and 29.4% and 19.1% compared with StringTie2 and Scallop on transcripts with 2–3 exons, 4–6 exons, and 7 or more exons, respectively.

Supplementary Note 5: Comparison on assembling transcripts at different expression levels. For each cell or sample, we first use Salmon[5] to quantify it using the reference transcriptome (Ensembl GRCh38.104 for human, Ensembl GRCm38.104 for mouse); we then collect all multi-exon transcripts (in reference annotation) with quantified abundance larger than a threshold (TPM $\geq$ 0.1) and divide them into three equal subsets corresponding to low, middle, high expression levels. Thus, we have three subsets of multi-exon transcripts for each cell or sample, and we use these three subsets as ground-truth for evaluation. The assembled multi-exon transcripts by each method will be evaluated against each of the ground-truth subset using gffcompare. Note that, when evaluated against one ground-truth subset, say the middle level, an assembled transcript will be classified as not matching (i.e., false transcript) if it is not in this middle-level ground-truth subset, even if it exists in other two ground-truth subsets. This will result in low precision but we think still reflect the *relative* performance of the compared methods. To get the highest sensitivity, the minimum coverage of all three methods are set as 0.001.

The results are illustrated in Supplementary Figure 9. While Scallop2 achieves higher sensitivity than StringTie2 and Scallop at all expression levels, the largest improvements are obtained at low expression levels. Averaged over 561 Smart-seq3 single cells, Scallop2 identifies 13.0% and 28.7% more matching multi-exon transcripts than StringTie2 and Scallop at low level, 8.9% and 19.3% more at middle level, and 1.6% and 5.1% more at high level. Averaged over 10 Illumina RNA-seq samples, Scallop2 identifies 77.0% and 17.1% more matching multi-exon transcripts than StringTie2 and Scallop at low level, 43.1% and 8.5% more at middle level, and 12.6% and 2.8% more at high expression level. Compared at the same level of sensitivity, Scallop2 keeps the best adjusted precision at all expression levels on all datasets. Averaged over 561 cells in two Smart-seq3 datasets, Scallop2 improves 120.2% and 109.7%, 111.5% and 100.7%, and 67.0% and 49.3%, in adjusted precision compared with StringTie2 and Scallop at low, middle, and high expression levels, respectively. Averaged over 10 Illumina RNA-seq samples, Scallop2 improves 21.9% and 15.1%, 50.7% and 25.2%, and 98.5% and 41.7% in adjusted precision compared with StringTie2 and Scallop at low, middle, and high expression levels, respectively.

Supplementary Note 6: Comparison of assembly accuracy with varying parameters. All three methods (StringTie2, Scallop, and Scallop2) support specifying two parameters, the minimum coverage threshold, to control the minimum expression abundance of a predicted transcript, and the minimum gap, to control the minimum gap of read mappings to start a gene loci for assembly. The default values of the minimum coverage used by StringTie2, Scallop, and Scallop2 are 1.0, 1.0 and 1.5, respectively. The default values of minimum gap used by StringTie2, Scallop, and Scallop2 are 50, 50, 100, respectively. Since highly expressed transcripts are easier to assemble, adjusting the parameter of the minimum coverage provides a trade-off of sensitivity and precision. In general, the sensitivity decreases while the precision increases when the minimum coverage gets higher. We run these three methods on different thresholds and draw the precision-sensitivity curve to see their capability to balance sensitivity and precision.

The results are illustrated in Supplementary Figures 10-12. The sensitivity-precision curves (Supplementary Figure 10) for Scallop2 were the highest for all datasets over all values of minimum gap. This indicates the improvement of Scallop2 over other two methods in a broad range of selected parameters. Also, at the same level of sensitivities Scallop2 constantly obtains higher adjusted precision than StringTie2 and Scallop over all combinations of parameters (Supplementary Figures 11–12).

Supplementary Note 7: Case study. To illustrate the effectiveness of the algorithmic heuristics used in Scallop2, we showcase two examples in Supplementary Figure 13 and describe how the Scallop2 algorithms lead to the correct assemblies. We use IGV[6] to visualize reads and assembled transcripts. The example BAM file using in the case study is a demultiplexed Smart-seq3 RNA-seq data with cell barcode AAGAGACGCATGCTTC in HEK293T dataset. The GTF files compared in the case study are reconstructed transcripts assembled by the three methods (StringTie2, Scallop, and Scallop2) using the example BAM file as input. To illustrate the ability of determines false starting/ending vertices, we visualize both full-length and non-full-length transcripts assembled by Scallop2.

In Supplementary Figure 13a, we show an example where there are gaps (i.e., regions not covered by any reads) in the 5th and the 6th exons. Both Scallop and StringTie2 interpret them as boundary exons and consequently the predictions are not full-length transcripts. The bridging algorithm used in Scallop2 bridges these multi-end reads (such as orange ones) as a single-end phasing path, and the bridged path span the 5th and the 6th exon which fills the gaps and also (partially) corrects the coverage of these two exons. Bridging algorithm helps to fill the gap using long-range information from paired-end/multi-end reads. Scallop2 therefore predicts only one transcript and it is a matching transcript.

In Supplementary Figure 13b, we show an example where Scallop2 successfully determines false starting/ending vertices. Both Scallop and StringTie2 assemble two transcripts. However, the paired-end/multi-end information (see the long gray thin lines which indicate paired-end reads not junctions) suggest that the two boundary exons (with highest coverage) are actually false boundary exons. The algorithms used in Scallop2 successfully determine them as false vertices and consequently label them as non-full-length transcripts.

Supplementary Note 8: Effectiveness of determining false starting/ending vertices. Scallop2 determines false starting/ending vertices in the splicing graph, and classifies the decomposed *s-t* paths as non-full-length transcripts if the path contains false vertex and otherwise as full-length transcripts. Here we evaluate the effectiveness of this approach. The hypothesis is that non-full-length transcripts exhibit low accuracy. To test this hypothesis, we use gffcompare to annotate each of the predicted transcript using the reference transcriptome. Gffcompare will label a transcript with one of the 15 classification codes (see documentation of GFF utilities[3]). Among them, code '=' represents complete, exact match of intron chains; this code is also used in this work to define if a predicted transcript is considered as *matching* or not. Code 'c' represents the predicted transcript is contained in a known transcript in the reference transcriptome; transcripts labeled as 'c' are most likely transcript fragments (i.e., non-full-length transcripts).

The distribution of the classifications of full-length transcript and non-full-length transcript assembled by Scallop2 is illustrated in Supplementary Figure 14. Non-full-length admits much lower accuracy (indicated by the much lower percentage of the '=' code). Additionally, a much larger portion of non-full-length transcripts are labeled as 'c', indicating that they are indeed transcript fragments. These two observations proves the effectiveness of the approach of determining false vertices.

Supplementary Algorithm

Supplementary Algorithm 1: Improved algorithm for phase-preserving graph decomposition.

Algorithm Consensus-Decompose (G', H, N)

01. for $k = 1 \to N$
02. while G' contains unsplittable vertices
03. if ($k = 1$): pick $\arg\min_{v:\ v \text{ is unsplittable}} z(v)$ and decompose it;
04. else: pick an unsplittable vertex randomly and decompose it;
05. end while
06. while G' contains splittable vertices
07. if ($k = 1$): pick $\arg\min_{v:\ v \text{ is splittable}} z'(v)$ and decompose it;
08. else: pick a splittable vertex randomly and decompose it;
09. goto line 02;
10. end while
11. while G' is not fully decomposed
12. pick a (trivial) vertex randomly and decompose it;
13. goto line 02;
14. end while
15. save the resulting *s*-*t* paths as P_k and their abundances as $f_k(p)$ for each $p \in P_k$;
16. end for
17. calculate $r(p) = \sum_{k=1}^{N} \delta(p \in P_k)$ for every $p \in \cup_{k=1}^{N} P_k$, and δ represents the indication function;
18. return $P = \{p \in \cup_{k=1}^{N} P_k \mid r(p) \geq N/2\}$ and $f(p) = \sum_{k=1}^{N} f_k(p)/r(p)$ for each $p \in P$;

References

[1] Dobin, A. et al. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* 29, 15–21 (2013).

[2] Kim, D., Langmead, B. & Salzberg, S. HISAT: a fast spliced aligner with low memory requirements. *Nat Methods* 12, 357–360 (2015).

[3] Pertea, G. & Pertea, M. GFF utilities: GffRead and GffCompare. *F1000Research* 9 (2020).

[4] Shao, M. & Kingsford, C. Accurate assembly of transcripts through phase-preserving graph decomposition. *Nat. Biotechnol.* 35, 1167–1169 (2017).

[5] Patro, R., Duggal, G., Love, M. & Kingsford, C. Salmon provides fast and bias-aware quantification of transcript expression. *Nat Methods* 14, 417–419 (2017).

[6] Robinson, J. et al. Integrative genomics viewer. *Nat. Biotechnol.* 29, 24–26 (2011).