Bioinformatics, 34(10), 2018, 1697–1704
doi: 10.1093/bioinformatics/btx839
Advance Access Publication Date: 29 December 2017
Original Paper

Sequence analysis

ChopStitch: exon annotation and splice graph construction using transcriptome assembly and whole genome sequencing data

Hamza Khan*,†, Hamid Mohamadi†, Benjamin P. Vandervalk, Rene L. Warren, Justin Chu and Inanc Birol*

Canada's Michael Smith Genome Sciences Centre, British Columbia Cancer Agency, Vancouver, BC V5Z 4S6, Canada

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

Associate Editor: Bonnie Berger

Received on January 25, 2017; revised on December 5, 2017; editorial decision on December 26, 2017; accepted on December 28, 2017

Abstract

Motivation: Sequencing studies on non-model organisms often interrogate both genomes and transcriptomes with massive amounts of short sequences. Such studies require *de novo* analysis tools and techniques, when the species and closely related species lack high quality reference resources. For certain applications such as *de novo* annotation, information on putative exons and alternative splicing may be desirable.

Results: Here we present ChopStitch, a new method for finding putative exons *de novo* and constructing splice graphs using an assembled transcriptome and whole genome shotgun sequencing (WGSS) data. ChopStitch identifies exon-exon boundaries in *de novo* assembled RNA-Seq data with the help of a Bloom filter that represents the *k*-mer spectrum of WGSS reads. The algorithm also accounts for base substitutions in transcript sequences that may be derived from sequencing or assembly errors, haplotype variations, or putative RNA editing events. The primary output of our tool is a FASTA file containing putative exons. Further, exon edges are interrogated for alternative exon-exon boundaries to detect transcript isoforms, which are represented as splice graphs in DOT output format.

Availability and implementation: ChopStitch is written in Python and C++ and is released under the GPL license. It is freely available at https://github.com/bcgsc/ChopStitch.

Contact: hkhan@bcgsc.ca or ibirol@bcgsc.ca

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The introduction of RNA-Seq almost a decade ago has provided researchers with a new and powerful technique for profiling the transcriptome of an organism (Wang *et al.*, 2009). RNA-Seq, also known as whole transcriptome shotgun sequencing, is the application of next generation sequencing technologies to read the nucleotides of RNA molecules in a cell sample. By comparing RNA-Seq reads to a reference genome, researchers are able to discover novel genes and isoforms, quantify gene and isoform expression levels, identify gene fusions, delineate exon/intron boundaries and characterize alternative splicing behaviour (Conesa *et al.*, 2016). However, the analysis of RNA-Seq is not without significant challenges. In contrast to whole genome shotgun sequencing (WGSS), RNA-Seq data represents a snapshot in time that varies across cell type and experimental conditions, and the abundances of RNA molecules can vary across several orders of magnitude. In addition to these considerations, the relatively short read lengths produced by leading next generation sequencing technologies can result in ambiguities during the alignment or *de novo* assembly of reads, which renders the task of reconstructing full-length transcripts intractable.

While current tools for RNA-Seq analysis perform well for predicting individual features of transcripts such as exons and splice sites, there is typically low agreement across tools when combining these components into full-length transcript predictions (Steijger *et al.*, 2013). The difficulties of the transcript reconstruction problem are further exacerbated by the prevalence of alternative splicing in eukaryotes. For instance, alternative splicing is believed to occur in more than 90% of multi-exon human genes (Douglas and Wood, 2011).

While the potential goals and applications of RNA-Seq analysis are broad (Conesa *et al.*, 2016), this paper focuses on the problems of exon prediction and splice graph prediction *de novo*, specifically in the case of organisms that lack reference genomes. Many approaches have previously been developed for predicting splicing behaviour using a reference genome or a reference set of gene models. For instance, Splicegrapher (Rogers *et al.*, 2012) constructs splice graphs by mapping expressed sequence tags (EST) and RNA-Seq reads onto a set of input gene models. More frequently, RNA-Seq tools use alignments of RNA-Seq reads to a reference genome as their primary input, where such alignments are generated by a splice-aware aligner such as TopHat2 (Kim *et al.*, 2013) or HISAT2 (Kim *et al.*, 2015). Two examples of such alignment-based approaches are Cufflinks (Trapnell *et al.*, 2010), which assembles and quantifies transcripts using the principle of parsimony, and StringTie (Pertea *et al.*, 2015), which assembles and quantifies transcripts using a network flow algorithm. Statistical tools like JunctionSeq (Hartley and Mullikin, 2016) can help detect, identify and characterize differential usage of specific exons or splice junctions, but rely on annotation files. In addition to analyzing read alignments, gene-prediction tools such as Augustus (Stanke *et al.*, 2006) can also incorporate evidence from the reference sequence itself, such as translation start/stop sites and acceptor/donor sites, in order to improve the confidence of their predictions.

Although comparison of RNA-Seq data to a reference genome has many useful applications, RNA-Seq data can also be highly valuable for the study of organisms that lack a reference genome (e.g. Wickett *et al.*, 2014; Birol *et al.*, 2015), as well as for cancer studies, where genomes may differ significantly from the reference. In comparison to reference-based methods, *de novo* RNA-Seq analyses introduce further challenges. While *de novo* transcriptome assemblers such as Trans-ABySS (Robertson *et al.*, 2010), Trinity (Grabherr *et al.*, 2011), Oases (Schulz *et al.*, 2012) and SOAPdenovo-Trans (Xie *et al.*, 2014) can be used to assemble transcript sequences directly from RNA-Seq reads, in practice these assemblers generate highly redundant collections of transcript *fragments* rather than complete transcripts (Conesa *et al.*, 2016), greatly complicating downstream analyses. Other *de novo* assembly method such as KISSPLICE(Sacomoto *et al.*, 2012), Bridger(Chang *et al.*, 2015) and BinPacker(Liu *et al.*, 2016) will generate splice graphs during assembly to help sensitivity and specificity of transcript reconstruction, however no genomic information is considered. As such, if only one transcript isoform is expressed, no splice junctions can be identified.

In the absence of a reference genome, transcript quantification can be performed by mapping the reads back to the assembled transcript fragments and counting aligned reads with software such as RSEM (Li and Dewey, 2011). Functional annotation of *de novo*-assembled transcripts can be performed using homology-based methods such as Blast2GO (Conesa *et al.*, 2005). Another example of a homology-based annotation method is LEMONS (Levin *et al.*, 2015), which identifies splice sites by aligning RNA-Seq reads to a reference set of peptides built from *H.sapiens* and eight model organisms.

Here we present ChopStitch, a novel method for predicting exon boundaries and splice graphs from *de novo* assembled transcript sequences, without reliance on a reference genome. ChopStitch operates by performing a *k*-mer based comparison between *de novo*-assembled transcript sequences and raw whole genome shotgun sequencing (WGSS) reads from the same organism. As genome sequencing represents a significant additional cost beyond that of RNA-Seq, we expect that our method will be best suited to research groups undertaking both transcriptome analysis and genome analysis for the same organism. In such a scenario, our method decouples RNA-Seq analysis from the costly prerequisite of building a reference-grade genome. To assess ChopStitch, we compare its performance against Augustus, Cufflinks, StringTie and LEMONS, using publicly available *C.elegans* and *H.sapiens* RNA-Seq datasets. We demonstrate that ChopStitch performs competitively with state-of-the-art reference-based methods, despite its lack of access to a reference genome.

2 Materials and methods

ChopStitch is implemented in C++ and Python, and consists of three utilities: CreateBloom, FindExons and MakeSplicegraph, for creating a Bloom filter, finding putative exons and constructing splice graphs, respectively. The output is a FASTA file of putative exons and a DOT (graph description language) file of splice graphs. ChopStitch uses several analysis tools to perform certain unit operations including the following.

2.1 ntHash

To build Bloom filters (see Section 2.3) ChopStitch uses the ntHash algorithm (Mohamadi *et al.*, 2016) to efficiently hash consecutive *k*-mers in a sequence. This hashing function is a recursive function, *f*, in which the hash value of the current *k*-mer, H_i, in the read sequence *r* is derived from the hash value of the previous *k*-mer, H_{i-1}, in *r*:

$$H_i = rol^1 H_{i-1} \oplus rol^k h(r[i-1]) \oplus h(r[i+k-1]) \tag{1}$$

where $\oplus$ is XOR operation, *rol* is left rotation, and *h* is a pre-designed 64-bit random seed table for DNA bases. This computation is started for the initial *k*-mer in the DNA/RNA sequence using the base function

$$H_0 = rol^{k-1} h(r[0]) \oplus rol^{k-2} h(r[1]) \oplus \ldots \oplus h(r[k-1]) \tag{2}$$

ntHash has a time complexity of $O(k+l)$ as compared to the $O(kl)$ complexity of regular hash functions, where *k* is the *k*-mer size and *l* is the length of sequence being hashed. ntHash has been shown to have significant speed improvement over traditional approaches, while maintaining a near-ideal hash value distribution (Mohamadi *et al.*, 2016).

2.2 ntCard

ntCard (Mohamadi *et al.*, 2017) is a streaming algorithm for estimating the frequencies of *k*-mers in genomics data. It first computes a 64-bit hash value for each *k*-mer in a sequence using ntHash, and selects a desired length of prefix and suffix bits termed *sample* (*s*) and *resolution* bits (*r*), respectively. The parameter *s* is used to reduce a dataset by $1/2^s$ on average, to prevent overpopulation of the counter space of size 2^r. The resolution bits *r* of subsampled hashes are then used to build a reduced representation multiplicity table describing the sample distribution. Finally, it uses a statistical model to reconstruct the population distribution from the sample distribution. Cardinality estimates of ntCard are used to automatically tune runtime parameters for Bloom filter sizes.

2.3 Implementation

2.3.1 Primary and secondary Bloom filters

A Bloom filter (Bloom, 1970) is a space-efficient probabilistic data structure that is used to test the set membership of an element. It has fast look-up times and a manageable false positive rate, and has been widely used in recent bioinformatics applications (Chu *et al.*, 2014; Melsted and Pritchard, 2011; Mohamadi *et al.*, 2015; Salikhov *et al.*, 2014). It can be described as a bit array of m bits, associated with φ hash functions, which map elements to positions in the bit array. The false positive rate in a Bloom filter can be estimated by:

$$\mathrm{FPR} = \left(1 - \left(1 - \frac{1}{m}\right)^{\varphi n}\right)^{\varphi} \tag{3}$$

Bloom filters provide an efficient way of cataloguing k-mers in genomic sequences for operations such as indexing and querying. In ChopStitch, all possible sub-strings of length k from the paired-end whole genome reads are hashed using ntHash and loaded into a Bloom filter. The optimal size of the Bloom filter is determined by the equation:

$$m = -\frac{n}{\ln^2(2)} \times \ln(\mathrm{FPR}) \tag{4}$$

ChopStitch uses ntCard to estimate the number of distinct k-mers, whereas the target FPR is specified by the user. Once the optimal size of the Bloom filter has been calculated, the optimal number of hash function can be determined with the equation:

$$\varphi = \frac{m}{n}\ln 2 \tag{5}$$

To remove the majority of k-mers caused by sequencing errors, ChopStitch discards all k-mers that occur only once in the data. This is performed using a two-level data structure called a cascading Bloom filter (Salikhov *et al.*, 2014), which consists of: (i) the primary Bloom filter (pbf), which stores k-mers that occur once or more in the data, and (ii) a secondary Bloom filter (sbf), which stores k-mers that occur twice or more in the data. To add a k-mer to the cascading Bloom filter, ChopStitch first checks to see if it is present in the pbf. If the k-mer is not present in the pbf, it inserts the k-mer into the pbf. If the k-mer is already present in the pbf, the k-mer is inserted into the sbf. After all k-mers have been passed through the cascading Bloom filter, the sbf contains the full set of k-mers that occur twice or more in the data. At this stage the pbf is discarded and the sbf is used as input for the downstream steps of ChopStitch.

Although not all k-mers with multiplicity of one would be error k-mers, and there would be error k-mers with multiplicity of two or more, this cascading approach is very useful as has been demonstrated before (Jackman *et al.*, 2017; Salikhov *et al.*, 2014; Vandervalk *et al.*, 2014, 2015). We also note that, as a result of using the cascading Bloom filter, genic regions with a WGSS read coverage of less than two would be omitted from our analyses.

2.3.2 Detecting exon–exon junctions

The k-mer content of assembled transcripts is interrogated in the secondary Bloom filter representing the whole genome shotgun data to detect exon-exon junctions. We note that all k-mers internal to exons would ideally be present in the secondary Bloom filter (sbf), while k-mers overlapping an exon-exon junction will be absent (Fig. 1). In an ideal situation, the stretch of absent k-mers at an exon-exon junction would be exactly equal to k-1. However, considering the FPR of the sbf, this value can be adjusted by a certain number of k-mers, which in ChopStitch, is governed by the *leniency* parameter. We define leniency as the number of k-mers that can be ignored while counting a k-1 absent k-mer stretch at a given position. Given the fpr and k, this can be calculated as:

$$\text{leniency} = \text{fpr} \times (k-1) \times \text{leniency factor} \tag{6}$$

In the Equation (6), the leniency factor dictates the level of leniency, and can be provided by the user as an input (default = 10).

2.3.3 Detecting and correcting sequencing errors and RNA editing events

The fidelity of a high-throughput sequencing dataset can be affected by several factors including sequencing errors, which would result in single or multiple base differences between the genome and transcriptome datasets of an individual. This would increase false positive predictions for ChopStitch as a single or multiple base difference between the transcripts, and the corresponding genome would return a k-1 or larger stretch of absent k-mers. A similar situation would be encountered in the case of a single nucleotide polymorphism (SNPs) or RNA editing event. Therefore, it is crucial to include reliable approaches for monitoring and reducing false positives while detecting exon-exon junctions.

In order to detect such events, the first k-mer found to be absent in the sbf after a series of present k-mers is scrutinized by substituting its last nucleotide in turn with each of the three alternate nucleotide bases and testing the presence of the resulting k-mers in the sbf. One out of the three resulting k-mers should ideally be present in the

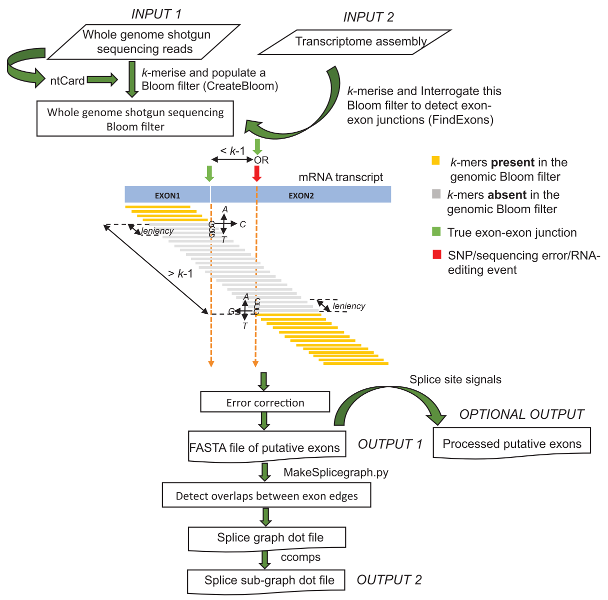

INPUT 1
Whole genome shotgun sequencing reads
INPUT 2
Transcriptome assembly
ntCard
k-merise and populate a Bloom filter (CreateBloom)
Whole genome shotgun sequencing Bloom filter
k-merise and Interrogate this Bloom filter to detect exon-exon junctions (FindExons)
< k-1
OR
mRNA transcript
EXON1
EXON2
leniency
> k-1
A
C
G
T
leniency
k-mers present in the genomic Bloom filter
k-mers absent in the genomic Bloom filter
True exon-exon junction
SNP/sequencing error/RNA-editing event
Splice site signals
Error correction
OPTIONAL OUTPUT
FASTA file of putative exons
OUTPUT 1
Processed putative exons
MakeSplicegraph.py
Detect overlaps between exon edges
Splice graph dot file
ccomps
Splice sub-graph dot file
OUTPUT 2

Fig. 1. ChopStitch workflow: After constructing the genomic Bloom filter, ChopStitch interrogates transcript sequences to find putative exons. It then finds exons with overlapping edges and constructs a splicegraph in DOT format. Graphviz ccomps is used to find sub-graphs. ChopStitch also detects putative exons smaller than the size of k-mer as illustrated in the figure: The stretch of absent k-mers is greater than k-1. The 3-sided arrows show the scrutiny process towards the beginning and end of the absent k-mer stretch

sbf, if the original *k*-mer had a sequencing error, a SNP or an RNA editing event at its end. Owing to the false positive rate of a Bloom filter, this step is repeated 'leniency' number of times for the next few *k*-mers in order to ascertain a true prediction. Although this step significantly reduces the number of false positives of our tool, it is advisable to use ChopStitch on the genome and transcriptome of the same individual to minimize single nucleotide differences due to SNPs. ChopStitch also has a post-processing functionality, that could be used by providing a list of conserved splice donor and acceptor sites for the species of interest. Using the genomic Bloom filter, it extends the ends of the obtained putative exons and looks for the presence of donor and acceptor sites for identifying exact splice sites (Supplementary Fig. S11). In ChopStitch, if an assembled transcript is not predicted to have any internal exon–exon junctions, it is not considered for further analysis. However, users have the option to keep these transcripts (by setting the optional parameter –allexons). It is to be noted that these transcripts might be incomplete single exons which might add to false positive predictions. The option also discards terminal exons that have only one exon–exon junction prediction.

2.3.4 Detecting putative exons smaller than *k*

Exons smaller than the *k*-mer size would return a stretch of absent *k*-mers equal to a length of [(*k*-1) + size of the small exon]. This property helps in identifying smaller exons that would otherwise have been missed (Fig. 1). A true exon-exon junction followed by one or more single base differences due to an RNA-editing event, a SNP or a sequencing error at a distance of less than *k*-1 would also return a similar profile of absent *k*-mers. As soon as a *k*-mer is found present after a stretch of absent *k*-mers of length greater than *k*-1, 'leniency' number of *k*-mers before this *k*-mer are scrutinized for detecting and correcting single base differences, as explained in section 2.3.3. The only difference is that, while in the former case substitutions are tested starting at the last base of the scrutinized *k*-mer, substitutions start from the first base in the latter case. If the scrutinized junctions in the absent *k*-mer stretch return true, then the smaller exon is outputted as a substring of the absent *k*-mer stretch ranging from its start to [length of the stretch– $(k - 1)$].

2.3.5 Splice graph construction

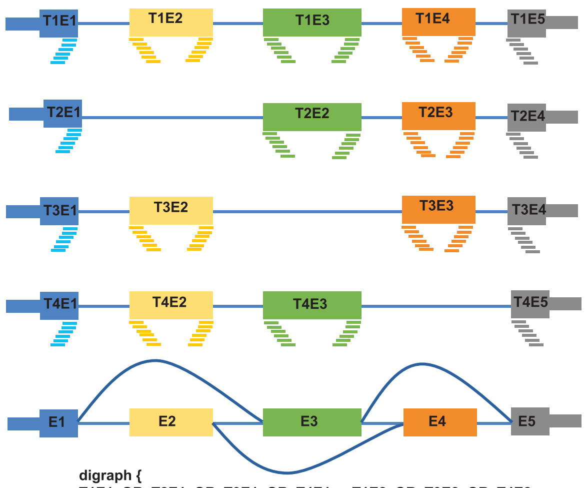
k-mers are retrieved from the edges of putative exons for a length equal to the leniency factor. These *k*-mers are cross-examined against each other by a Python script to find overlapping edges between putative exons belonging to different transcript isoforms (Fig. 2). This information is used to collect exons in a single node, and generate splice graphs for the complete set of transcripts, which is then stored as a single graph in DOT format. Graphviz ccomps function (Ellson *et al.*, 2004) is then used to output subgraphs from this file.

These subgraphs depict putative splice variants of a gene while offering a representation of observed transcript isoforms and laying out their splicing patterns. These can be visualized using DOT visualization tools such as Graphviz (Ellson *et al.*, 2004) and Gephi (Bastian *et al.*, 2009).

3 Results

3.1 Dataset

We have used experimental DNA and RNA sequencing datasets of *H.sapiens* and *C.elegans* (Table 1) to test and compare the performance of ChopStitch and benchmark our results against that of leading tools. The *H.sapiens* datasets are from a 1000 Genomes Project individual, NA19238, and the *C.elegans* datasets were obtained for the N2 strain.



```
digraph {
T1E1_OR_T2E1_OR_T3E1_OR_T4E1 -> T1E2_OR_T3E2_OR_T4E2;
T1E1_OR_T2E1_OR_T3E1_OR_T4E1 -> T1E3_OR_T2E2_OR_T4E3;
T1E2_OR_T3E2_OR_T4E2 -> T1E3_OR_T2E2_OR_T4E3;
T1E2_OR_T3E2_OR_T4E2 -> T1E4_OR_T2E3_OR_T3E3;
T1E3_OR_T2E2_OR_T4E3 -> T1E4_OR_T2E3_OR_T3E3;
T1E4_OR_T2E3_OR_T3E3 -> T1E5_OR_T2E4_OR_T3E4_T4E5;
T1E4_OR_T2E3_OR_T3E3 -> T1E5_OR_T2E4_OR_T3E4_T4E5;
}
```

Fig. 2. Constructing the splice graph DOT file: In the above figure, T_i and E_i represent the transcript and exon ids belonging to different protein coding transcript isoforms. *k*-mers are shown as colored lines towards the edge of the putative exon. Putative exons which share the same *k*-mer spectra are shown by the same color and are represented as a single node in the DOT file with their headers concatenated with the string '_OR_' (Color version of this figure is available at *Bioinformatics* online.)

Table 1. Dataset specification

Organism	Library strategy	Accession ID	Read length	# Reads
H.sapiens	WGSS	ERR309932	250 bp	228 489 950 000
C.elegans	WGSS	DRR008444	110 bp	68 621 900
H.sapiens	RNA-Seq	ERR356374	50 bp	43 166 926
C.elegans	RNA-Seq	SRR2537190	50 bp	23 983 224

3.2 Experimental setup

We compared our method with Augustus 3.2.2, LEMONS, Cufflinks 2.2.1 and StringTie 1.3.3b. While LEMONS, like ChopStitch, is a tool for the identification of splice junctions in transcriptomes of organisms lacking reference genomes, Cufflinks and StringTie require a reference genome for predicting putative exons. LEMONS utilizes a BLAST based approach, and exploits the evolutionary conservation of both splice junction recognition signals and exon/intron boundary positions to predict splice-junctions in transcripts in the absence of a reference genome. It outputs separate results for each reference database and a merged result based on all databases. Augustus uses generalized hidden Markov models and probabilistic models to predict genes in eukaryotic genome sequences. On the other hand, Cufflinks and StringTie use a BAM file of RNA-Seq reads aligned against a reference genome (using TopHat 2.1.1.) as an input to assemble and quantitate full-length transcripts representing multiple splice variants for every gene locus. Owing to a high percentage of small introns in the *C.elegans* genome, we set the TopHat parameters –min-coverage-intron, –min-segment-intron and -i to 30 for *C.elegans* and 50 for *H.sapiens* (Steijger *et al.*, 2013). For Cufflinks, the –min-intron-length parameter was set to 30 for *C.elegans* and 50 for *H.sapiens*. All the tools were run with 32 threads and default parameters (except TopHat and Cufflinks) on an isolated machine with 2.5 TB RAM and 128 Xeon E7-8867 CPU cores.

For input to ChopStitch and LEMONS, we generated *de novo* transcriptome assemblies of the *H.sapiens* and *C.elegans* RNA-Seq datasets (Table 1) using Trans-ABySS v1.5.5 (Robertson *et al.*, 2010). We ran the Trans-ABySS assemblies using *k*-mer sizes of 25, 35 and 45, and used the default values for all other parameters. To merge the assemblies generated at the three different *k*-mer sizes, we used the transabyss-merge utility in the Trans-ABySS package, and excluded contigs shorter than 100 base pairs from the final output. In addition, we also assessed the quality of the Trans-ABySS assemblies with rnaQUAST (Bushmanova *et al.*, 2016) (Supplementary Table S1A and B).

3.3 Evaluation criteria for exon–exon boundary detection

We downloaded exon annotations from BioMart Ensembl for *H.sapiens* (Download date: July 21, 2016) and *C.elegans* (Download date: December 19, 2015). Putative exons from all tools were assessed by aligning them against reference exons from Ensembl using BLAST with 95 percent query coverage, 95 percent sequence identity, and a difference of five base pairs between the subject's and query's actual length. Putative exons that returned a hit to a reference exon while following these criteria were considered to be True Positives (TP), while putative exons that did not return a hit were considered as False Positives (FP). Precision was calculated as the ratio of the number of true positives to the total number of putative exons. False negatives (FN) were defined as those reference exons that did not have a BLAST hit to a putative exon. This was calculated by subtracting reference exons that had a hit with a putative exon, with the total number of reference exons. To define the recall of these predictions, we divided TP with the sum of TP and FN. As RNA-Seq captures the presence and quantity of RNA in a sample at a given moment in time, the number of transcripts assembled from such datasets will be less in comparison to the number of transcripts in comprehensive set of gene models available in reference databases. This is reflected in Supplementary Table S1A and B, where the database coverage for both datasets by our assemblies is low. Because we are comparing our results to the complete set of annotated exons, as opposed to previous studies, which processed RNA-seq data restricting comparisons to annotated exons that are expressed (Steijger *et al.*, 2013), these studies report higher recall rates with respect to the recall rates reported here. However, note that our recall rates would be consistently low for all tools; hence the metric would still be relevant to compare their relative performances.

For a more comprehensive analysis, we compiled a set of non-redundant reference and putative exons from all tools, and evaluated their ability in detecting exons at a distance of 0–5 base pairs from the exact exon–exon junction [Please refer Supplementary Note 4, Supplementary Table S6(a–c), S7(a–c), S8(a–c), S9(a–c) and Supplementary Figs S1–S4]. We refer the evaluation using non-redundant exons as non-redundant analysis, while analysis with normal exon-sets was referred to as redundant analysis.

3.4 ChopStitch performance

ChopStitch is competitive when compared to the state-of-the-art methods available for finding putative exons with and without the help of a reference genome as demonstrated in Figure 3A–E. From Figure 3B and C, and Supplementary Table S2b we can see that in *H.sapiens* at $k = 50$, $\mathrm{pbf}_{\mathrm{FPR}}$=0.16, $\mathrm{sbf}_{\mathrm{FPR}}$=0.01 and leniency = 10, ChopStitch finds non-redundant putative exons at a precision of 0.87 and a recall of 0.15. LEMONS has a precision and recall of 0.32 and 0.06 for merged results and 0.80 and 0.18 using the same species reference database, respectively. The results for StringTie and Cufflinks are similar to each other, with StringTie having a precision and recall of 0.71 and 0.16, and Cufflinks having a precision and recall of 0.63 and 0.16, respectively. Although Augustus had a large number of hits to reference exons, it returned an overall precision of 0.56 and a recall of 0.29 due to a higher number of FP predictions. Similar results were obtained in the case of *C.elegans* (Fig. 3B and C and Supplementary Table S3a and b). We observed that in this case, the performance of Cufflinks and StringTie depends on parameter tuning and drops with default parameters (Supplementary Fig. S12). ChopStitch exons also show a similar length distribution as compared to reference exons (Supplementary Fig. S10). We observed that the number of hits to reference exons and the recall increased for putative exons with -allexons output. However, as expected there is a decline in precision which might be due to incomplete transcript assemblies.

Owing to the Bloom filter FPR, ChopStitch's performance depreciates while finding exact boundaries, but significantly improves at a difference of two or more bases [Supplementary Tables S6(a–c), S7(a–c) and Supplementary Figs S1 and S2]. When comparing with non-redundant putative and reference exons, there was a slight increase in precision with the other tools [Supplementary Note 4, Supplementary Tables S8(a–c), S9(a–c) and Supplementary Figs S3 and S4]. Generally, all tools output a similar profile of exon types (5′, 3′, middle, singleton). Most tools were able to find more middle exons as compared to 5′, 3′ or single exons. We calculated the ratio of reference exons (of a particular type) with hits to putative exons to the total number of reference exons (of the same type) and found that the ratio for middle exons is a lot more comparable between tools, especially for *H.sapiens* [Supplementary Table S10(a–h) and Supplementary Fig. S5a and b].

For each combination of dataset and tool, we measured run time and peak memory usage. In Figure 3D and E, we observe that StringTie was the fastest tool with the lowest peak memory usage. While ChopStitch showed competitive run times and memory usage, LEMONS and Augustus were notably slower (4-fold slower on average) with high peak memory (4-time higher on average). ChopStitch's peak memory requirement is directly proportional to the size of the pbf, which is governed by the approximate number of distinct *k*-mers outputted by ntCard.

We also evaluated the effect of DNA and RNA-Seq sequencing coverage on the performance of ChopStitch (Supplementary Note 5, 6, Supplementary Table S11 and Supplementary Figs S6 and S7). With respect to DNA coverage, ChopStitch was able to generate comparable results with less than half the original WGS read data. For evaluating the effect of RNA-Seq coverage, we quantified transcript expression through RSEM (Li and Dewey, 2011) and calculated TPM (Transcripts Per Kilobase Million). We found that ChopStitch was able to detect exons for transcripts even with very low TPM scores.

We also evaluated ChopStitch's performance while using a reference genome (Supplementary Tables S12 and S13 and Supplementary Fig. S9). The results are almost similar to those using genomic reads, which speaks to the accuracy of WGS sequencing data. However, a slight decrease in precision is observed when using the reference instead of genomic reads. This is possibly due to

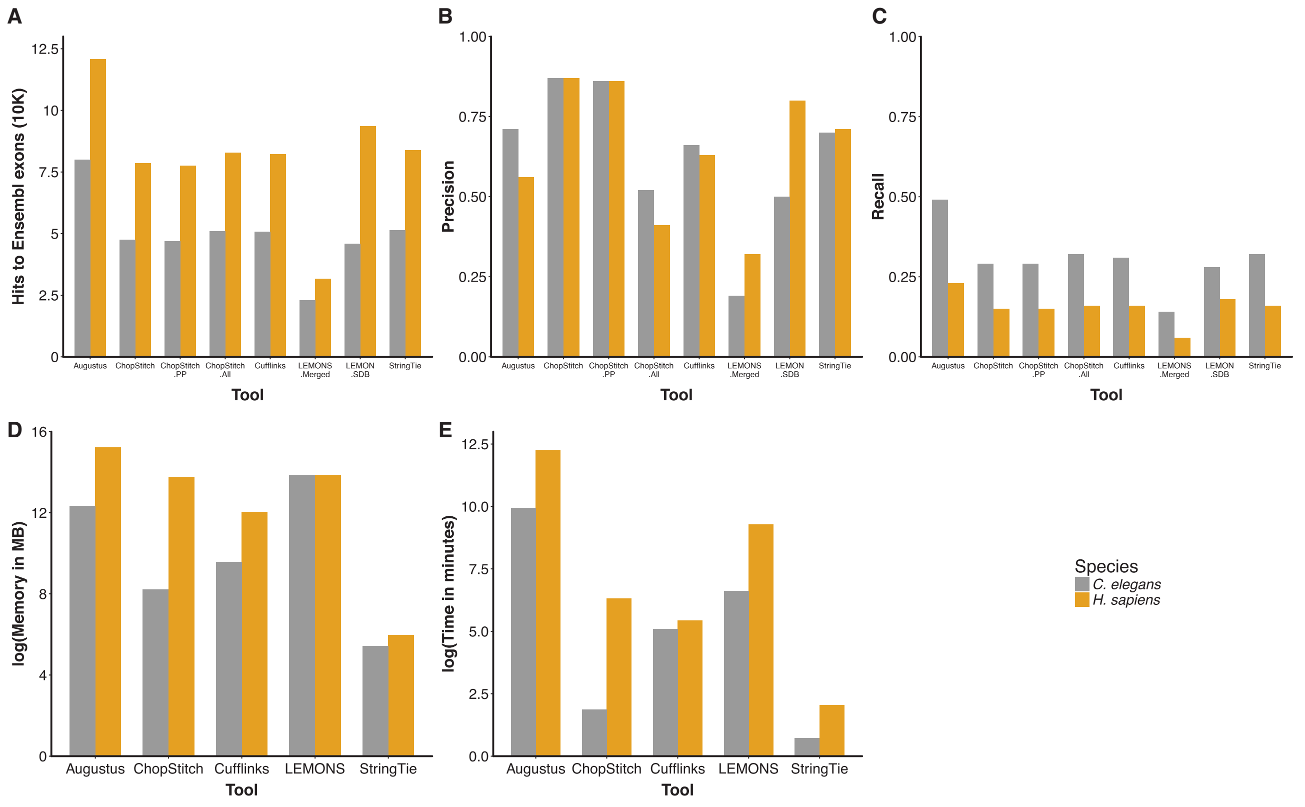


Fig. 3. Performance of ChopStitch compared to state-of-the-art exon prediction software tools (Non-redundant analysis). (**A**) For each tool, the number of BLAST hits returned against non-redundant Ensembl exons at a query coverage and sequence identity of 95% and a length difference of 5 base pairs between the query and the subject, (**B**) Precision comparison, (**C**) Recall comparison, (**D**) Comparing $\log_2$ of memory consumption in MB between tools, (**E**) Comparing $\log_2$ of time in minutes between tools (ChopStitch.PP denotes the results obtained after running the post-processing step in ChopStitch. ChopStitch.All are the results obtained after running ChopStitch with the –allexons option. LEMONS-Merged show results from the LEMONS merged.xls file. LEMONS-SDB show LEMONS results while using the same species reference database)

increased SNPs between the reference genome and assembled transcripts, which is expected given the mosaicism of the reference sequence set whereas base variation between genomic reads and assembled transcripts from the same individual/organism is not.

3.5 Effect of primary and secondary Bloom filter FPRs

We evaluated ChopStitch's performance with different false positive rates for pbf and sbf in order to optimize our tool. We measured the performance based on F1-scores, which can be used to measure the accuracy of a test in the statistical analysis of a binary classification. It considers both the precision and recall of a test and can be interpreted as a weighted average of both. In other words, it is the harmonic mean of precision and recall.

Figure 4 shows ChopStitch results for *C.elegans* on a range of FPR values for pbf and sbf. The F1-score was calculated while increasing sbf_{FPR} by 0.005 and pbf_{FPR} by 0.05 at each measurement. ChopStitch's performance seems to deteriorate with a small increase in sbf_{FPR} as compared to a much larger increase in pbf_{FPR}. This is due to the size of sbf being appreciably smaller than pbf, and many false positive *k*-mers are removed during the pbf to sbf reduction step. Besides, it is sbf that is used to interrogate *k*-mers in ChopStitch, and has a direct role in exon-exon junction prediction.

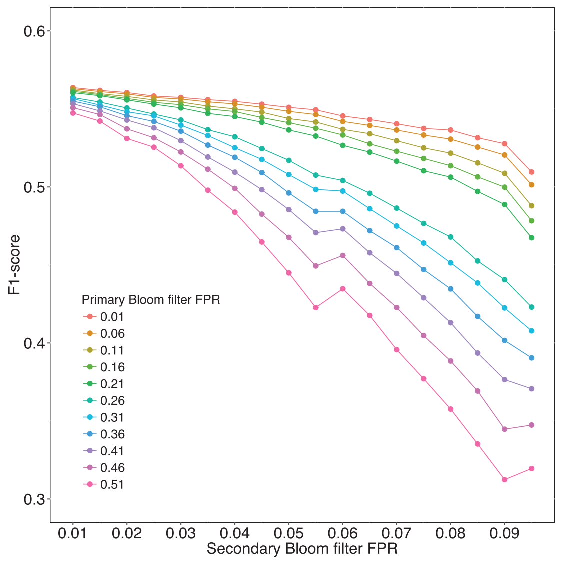


Fig. 4. ChopStitch results for *C.elegans* on different pbf and sbf FPR values. Performance deteriorates with a small increase in sbf_{FPR} compared to a much larger increase in pbf_{FPR}

3.6 Effect of leniency

We have also evaluated the effect of the leniency parameter on the performance of ChopStitch (Supplementary Tables S4 and S5). Broadly, on both *C.elegans* and *H.sapiens* datasets, we have observed a sharp increase in performance from a leniency factor of zero to five. On increasing it further, the performance seems to marginally increase, plateaus and deteriorates for latter increments. This marginal difference in performance could be explained by the fact

Table 2. Evaluation results for ChopStitch splice graphs

Organism	Evaluation approach	Precision
H.sapiens	*de novo*	0.97
C.elegans	*de novo*	0.95
H.sapiens	Reference-based	0.94
C.elegans	Reference-based	0.96
H.sapiens	Annotation-based	0.90
C.elegans	Annotation-based	0.98

that for these species, both the transcriptome and genome datasets belong to the same individual, and would have a low number of single nucleotide differences. We expect variation in ChopStitch results for different leniency values to increase with unmatched datasets, where there will be increased chances of single nucleotide differences between the genome and transcriptome of the organisms studied.

3.7 Assessment of putative splice graphs

Although ChopStitch is a *de novo* analysis tool, we found it relevant to compare the obtained splice graph with *de novo* and reference-based strategies.

3.7.1 De novo evaluation

Owing to the Bloom filter FPR, length of putative exons may differ from their actual length by a few nucleotides. However, we expect all putative exons that form a single node in a splice graph to be substrings of the longest putative exon in that node. As seen in Table 2, we found that 97% of 31 302 nodes and 95% of 11 264 nodes, present in the splice graphs for *H.sapiens* and *C.elegans*, followed this criterion.

3.7.2 Reference based evaluation

We aligned all putative exons predicted for *H.sapiens* and *C.elegans* against their reference genome and expect all putative exons that are part of a single node in a splice graph to align to approximately the same coordinates. We found that, 94 and 96% of the total nodes in the splice graphs for *H.sapiens* and *C.elegans* respectively, followed this criterion.

3.7.3 Annotation based evaluation

Subgraphs obtained after running Graphviz ccomps represent different transcript isoforms for a single gene. We expect all exons in a splice-subgraph to come from the same gene. Using Ensembl reference transcripts along with experimental WGSS data (Table 1) as inputs to ChopStitch, we found that 90% of 18 008 splice-subgraphs in *H.sapiens* and 98% of 15 087 splice-subgraphs in *C.elegans* to follow this criterion.

4 Discussion and conclusion

ChopStitch is a new method for finding putative exons *de novo* and constructing splice graphs using an assembled transcriptome and, for best results, corresponding whole genome shotgun sequencing data. It enables the identification of exon–exon boundaries in *de novo* assembled RNA-Seq data with the help of Bloom filters that represent the *k*-mer spectra of WGSS reads. The algorithm also compensates for base substitutions in transcript sequences that may correspond to sequencing or assembly errors, haplotype variations, or putative RNA editing events. It incorporates the state-of-the-art technologies ntHash and ntCard for a fast and efficient workflow. The primary output of our tool is a FASTA file containing putative exons. Further, *k*-mer content of the edges of putative exons are interrogated against other exons to detect and represent transcript isoforms, reported as splice graphs in DOT output format.

ChopStitch's ability to use unassembled genomic datasets reduces its reliance on the contiguity and completeness of a genome assembly for the annotation process. The post-processing algorithm can help find exact exon-exon junctions and is an avenue for future research. Our method could be used to annotate *de novo* transcriptome assemblies, find novel-exons in coding as well as non-coding mRNA transcripts, and search alternative mRNA splicing events in non-model organisms, thus exploring new loci for functional analysis and studying genes that were previously inaccessible.

Funding

We thank Genome Canada, Genome BC, and British Columbia Cancer Foundation for their financial support. The work is also partially funded by the National Institutes of Health under Award Number R01HG007182. The content of this work is solely the responsibility of the authors, and does not necessarily represent the official views of the National Institutes of Health or other funding organizations.

Conflict of Interest: none declared.

References

Bastian,M. *et al.* (2009) Gephi: an open source software for exploring and manipulating networks. *Icwsm*, **8**, 361–362.

Birol,I. *et al.* (2015) De novo transcriptome assemblies of rana (Lithobates) catesbeiana and *Xenopus laevis* tadpole livers for comparative genomics without reference genomes. *PLoS One*, **10**, 1–18.

Bloom,B.H. (1970) Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, **13**, 422–426.

Bushmanova,E. *et al.* (2016) rnaQUAST: a quality assessment tool for de novo transcriptome assemblies. *Bioinformatics*, **32**, 2210.

Chang,Z. *et al.* (2015) Bridger: a new framework for de novo transcriptome assembly using RNA-seq data. *Genome Biol.*, **16**, 30.

Chu,J. *et al.* (2014) BioBloom tools: fast, accurate and memory-efficient host species sequence screening using bloom filters. *Bioinformatics*, **30**, 3402.

Conesa,A. *et al.* (2005) Blast2go: a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics*, **21**, 3674–3676.

Conesa,A. *et al.* (2016) A survey of best practices for RNA-seq data analysis. *Genome Biol.*, **17**, 13.

Douglas,A.G.L. and Wood,M.J.A. (2011) RNA splicing: disease and therapy. *Brief. Funct. Genomics*, **10**, 151.

Ellson,J. *et al.* (2004) Graphviz and dynagraph-static and dynamic graph drawing tools. *Graph drawing software*, 127–148.

Grabherr,M.G. *et al.* (2011) Full-length transcriptome assembly from RNA-seq data without a reference genome. *Nat. Biotechnol.*, **29**, 644–652.

Hartley,S.W. and Mullikin,J.C. (2016) Detection and visualization of differential splicing in RNA-Seq data with JunctionSeq. *Nucleic Acids Res.*, **44**, e127.

Jackman,S.D. *et al.* (2017) ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter. *Genome Res.*, **27**, 768–777.

Kim,D. *et al.* (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol.*, **14**, R36.

Kim,D. *et al.* (2015) HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods*, **12**, 357–360.

Levin,L. *et al.* (2015) LEMONS – a tool for the identification of splice junctions in transcriptomes of organisms lacking reference genomes. *Plos One*, **10**, 15.

Li,B. and Dewey,C.N. (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics*, **12**, 323.

Liu,J. *et al*. (2016) Binpacker: packing-based de novo transcriptome assembly from RNA-seq data. *PLoS Comput. Biol*., **12**, e1004772.

Melsted,P. and Pritchard,J.K. (2011) Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinformatics*, **12**, 333.

Mohamadi,H. *et al*. (2015) DIDA: Distributed Indexing Dispatched Alignment. *PLoS One*, **10**, 1–14.

Mohamadi,H. *et al*. (2016) ntHash: recursive nucleotide hashing. *Bioinformatics*, **32**, 3492–3494.

Mohamadi,H. *et al*. (2017) ntCard: a streaming algorithm for cardinality estimation in genomics data. *Bioinformatics*, **33**, 1324.

Pertea,M. *et al*. (2015) StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nat. Biotechnol*., **33**, 290–295.

Robertson,G. *et al*. (2010) De novo assembly and analysis of RNA-seq data. *Nat. Methods*, **7**, 909–912.

Rogers,M.F. *et al*. (2012) SpliceGrapher: detecting patterns of alternative splicing from RNA-Seq data in the context of gene models and EST data. *Genome Biol*., **13**, R4.

Sacomoto,G.A. *et al*. (2012) K is s plice: de-novo calling alternative splicing events from RNA-seq data. *BMC Bioinformatics*, **13**, S5.

Salikhov,K. *et al*. (2014) Using cascading Bloom filters to improve the memory usage for de Brujin graphs. *Algorithms Mol. Biol*., **9**, 2.

Schulz,M.H. *et al*. (2012) Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. *Bioinformatics*, **28**, 1086–1092.

Stanke,M. *et al*. (2006) Augustus: ab initio prediction of alternative transcripts. *Nucleic Acids Res*., **34**, W435–W439.

Steijger,T. *et al*. (2013) Assessment of transcript reconstruction methods for rna-seq. *Nat. Methods*, **10**, 1177–1184.

Trapnell,C. *et al*. (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat. Biotechnol*., **28**, 511–515.

Vandervalk,B.P. *et al*. (2014). Konnector: connecting paired-end reads using a bloom filter de Bruijn graph. In: *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 51–58.

Vandervalk,B.P. *et al*. (2015) Konnector v2.0: pseudo-long reads from paired-end sequencing data. *BMC Med. Genomics*, **8**, S1.

Wang,Z. *et al*. (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet*., **10**, 57–63.

Wickett,N.J. *et al*. (2014) Phylotranscriptomic analysis of the origin and early diversification of land plants. *Proc. Natl. Acad. Sci. USA*, **111**, E4859–E4868.

Xie,Y. *et al*. (2014) Soapdenovo-trans: de novo transcriptome assembly with short RNA-seq reads. *Bioinformatics*, **30**, 1660–1666.