Yanagi: Transcript Segment Library Construction for RNA-Seq Quantification

Mohamed K. Gunady[1,2], Steffen Cornwell[4], Stephen M. Mount[2,3], and Héctor Corrada Bravo[1,2]

1 **Department of Computer Science, University of Maryland, College Park, Maryland, USA**
mgunady@cs.umd.edu
hcorrada@umiacs.umd.edu
2 **Center for Bioinformatics and Computational Biology, University of Maryland, College Park, Maryland, USA**
3 **Department of Cell Biology and Molecular Genetics, University of Maryland, College Park, Maryland, USA**
smount@umd.edu
4 **School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, Pennsylvania, USA**

Abstract

Analysis of differential alternative splicing from RNA-seq data is complicated by the fact that many RNA-seq reads map to multiple transcripts, and that annotated transcripts from a given gene are often a small subset of many possible complete transcripts for that gene. Here we describe Yanagi, a tool which segments a transcriptome into disjoint regions to create a segments library from a complete transcriptome annotation that preserves all of its consecutive regions of a given length L while distinguishing annotated alternative splicing events in the transcriptome. In this paper, we formalize this concept of transcriptome segmentation and propose an efficient algorithm for generating segment libraries based on a length parameter dependent on specific RNA-Seq library construction. The resulting segment sequences can be used with pseudo-alignment tools to quantify expression at the segment level. We characterize the segment libraries for the reference transcriptomes of Drosophila melanogaster and Homo sapiens. Finally, we demonstrate the utility of quantification using a segment library based on an analysis of differential exon skipping in Drosophila melanogaster and Homo sapiens. The notion of transcript segmentation as introduced here and implemented in Yanagi will open the door for the application of lightweight, ultra-fast pseudo-alignment algorithms in a wide variety of analyses of transcription variation.

1998 ACM Subject Classification I.1.2 Algorithms

Keywords and phrases RNA-Seq - Genome Sequencing - Kmer alignment - Transcriptome Quantification - Differential Alternative Splicing

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Messenger RNA transcript abundance estimation from RNA-Seq data is a crucial task in studies that seek to describe the effect of genetic or environmental changes on gene expression. Differential expression analysis over either genes or transcripts is used to find the set of genes or transcripts with different expression levels between conditions. Although there are many tools that can provide satisfying results at the gene level, transcript level analysis still faces major challenges that make the problem of transcript expression quantification harder.


© Mohamed K. Gunady, et al.;
licensed under Creative Commons License CC-BY

Over the years, various approaches have addressed the joint problems of (gene level) transcript expression quantification and differential alternative RNA processing. Much effort in the area has been dedicated to the problem of efficient alignment of reads to a genome or a transcriptome, since this is typically a bottleneck in the analytical processes that start with RNA-Seq reads and yield gene-level expression or differentially expressed transcripts. Among these approaches are alignment techniques such as bowtie [6], Tophat [16, 5], and Cufflinks [17], and newer techniques such as sailfish [10], RapMap [13], Kallisto [2] and Salmon [9], which provide efficient strategies that are much faster, but maintain comparable, or superior, accuracy.

In order to achieve faster alignment and quantification, the newer methods introduced novel approaches, such as alignment-free k-mer based quantification [Sailfish], quasi-mapping [RapMap], pseudo-alignment [Kallisto], or lightweight alignment [Salmon]. These methods all simplified the expected outcome of the alignment step, finding only sufficient read-alignment information required by the quantification step, utilizing k-mer counting to be the sufficient statistic built from the alignment step. In other words, given a transcriptome reference, an index of kmers is created and used to find a mapping between reads and the list of compatible transcripts based on each approach's definition of compatibility. The next step would be to resolve the ambiguity in reads that were mapped to multiple transcripts. Multi-mapping reads are common even assuming error free reads, due to shared regions produced by alternative splicing. The ambiguity in mapping reads is resolved using probabilistic models, such as the EM algorithm, to produce the abundance estimate of each transcript [8].

The presence of sequence repeats and paralogous genes in many organisms also creates ambiguity in the placement of reads. Moreover, the fact that alternatively spliced transcripts share most of their genomic region, greatly increases the portion of reads coming from these shared regions and consequently reads being multi-mapped becomes more frequent when aligning to enumerated transcripts. In fact, local splicing variations can be joined combinatorially to create a very large number of possible transcripts from many genes. An extreme case is the Drosophila gene Dscam, which can produce over 38,000 transcripts by joining less than 50 exons [19]. More generally, long-read sequencing indicates that although there are correlations between distant splicing choices [15], a large number of possible combinations is typical. Thus, standard annotations, which enumerate only a minimal subset of transcripts from a gene (e.g. [3]) are inadequate descriptions. Furthermore, short read sequencing, which is likely to remain the norm for some time, does not provide information for long-range correlations between splicing events.

In this paper, we propose a novel strategy that aims at constructing a set of segments that can be used in the read-alignment-quantification steps instead of the whole transcriptome without loss of information. Such a set of segments (a segment library) can fully describe individual events (primarily local splicing variation, but also editing sites or sequence variants) independently, leaving the estimation of transcript abundances as a separate problem. Here we introduce and formalize the idea of transcriptome segmentation, propose and analyze an algorithm for transcriptome segmentation, and present a tool called Yanagi, which implements this segmentation algorithm to build a segment library from a reference transcriptome based on the possible splicing variations. We show results from the application of Yanagi to reference transcriptomes of Drosophila melanogaster and Homo sapiens that characterize the resulting segment libraries. Since the segment libraries are amenable for usage with lightweight pseudo-alignment methods for segment quantification, we illustrate the utility of the segmentation approach using the differential analysis of exon skipping events across samples from two conditions of interest. We use simulation studies in Drosophila melanogaster

and Homo sapiens and show that this is a promising approach for this type of analysis.

2 Methodology

Exonic regions of a messenger RNA precursor can be combined differently through alternative splicing (AS) to form distinct isoforms. Alternative transcripts can be generated by AS proper (including exon skipping, mutual exon exclusion, intron retention, and alternative splice site use), alternative transcription start sites, and alternative 3' termini (sites of cleavage and polyadenylation). Combinations of these allow more complex events. A comprehensive treatment of the different types of splicing events can be found in [18]. Over 95% of human genes with multiple exons undergo AS [18]; consequently a majority of the coding genomic region is spliced into more than one isoform.

The goal of our approach is to segment the transcriptome into a set of disjoint regions (where disjointness is parameterized by a specific read length) without losing any possible transcriptome sub-sequence that may be sequenced in a given RNA-Seq experiment. Afterwards, we can pseudo-align reads into the set of segments and quantify abundance at the segment level for use in further downstream analysis. Consequently, our quantification pipeline can use available kmer-based pseudo-mapping or pseudo-alignment techniques over the set of segments generated by Yanagi from the transcriptome reference and generate counts for segments. The rest of this section describes Yanagi's algorithm for generating the segment library. We later discuss how it can be used for quantification purposes using differential analysis of exon skipping events as an illustrative use case.

2.1 Transcriptome Segments Properties

▶ **Definition 1.** Segment

A segment $seg(Exs, loc, w)$ is a genomic region of width w beginning at genomic location loc and spanning the sequence of consecutive exonic regions Exs. Exonic regions are considered consecutive if they are consecutively joined into at least one possible isoform.

▶ **Definition 2.** L-disjoint property.

The set of segments S is *L-disjoint* if and only if

$$width[overlap(seg_i, seg_j)] < L; \forall seg_i, seg_j \in S, i \neq j$$

That restricts any pair of *L-disjoint* segments to have an overlap region shorter than parameter L, corresponding to the read length of a specific RNA-Seq experiement. In other words no read of length at least L can be mapped to both segments of an *L-disjoint* segment pair, assuming error-free reads.

Given a reference transcriptome, a naive approach to generating such L-disjoint segments would be to use the set of exonic regions and junctions defined in the transcriptome and generate segments spanning each exonic region and junction. Specifically, a junction segment would be formed by spanning $L - 1$ positions from both sides of the junction and exon segments would simply include the genomic sequence of the exon that does not overlap any of the junction segments. Figure 1 (left) shows a simple exon skipping event using splicing graph representation [4] and the corresponding generated segments following that naive approach. This approach would successfully generate segments capturing all possible sequences required to map any read to the transcriptome. However this naive approach faces a few challenges.

First, exons that are shorter than parameter L are problematic. For instance, around 30% of the exons in the UCSC hg38 genome are shorter than 100bp (Illumina's common

paired-end read length). These short exons will make junction segments miss reads that span more than two of such short exons. Consider the example in Figure 1 (right) where the two exons E2, E3 of width k are both shorter than L, no segments will capture a read that span E1, E2, and E3 for instance.

Another related challenge is that the annotated exons are not strictly disjoint in the reference itself. Some annotated exons overlap due the use of alternative transcription start and end sites. Such challenges indicate that more careful choice of segments is necessary to guarantee the L-disjointness property, so we formalized an additional segment property.

First, denote $Txs(exs)$ as the set of annotated transcripts splicing exons $exs \in Exs$, and $Txs(seg)$ as the union of $Txs(exs)$ for exons exs included in segment seg. We can define a subsumption relationship between segments as $seg_1(Exs, loc, w) \succ seg_2(Exs, loc, w)$ if $Txs(seg_1) = Txs(seg_2)$ and $width(seg_1) > width(seg_2)$. With this relationship we can define the following property of a segment library.

▶ **Definition 3.** Max-spanning property.
$seg_1(Exs, loc, w) \succ seg_2(Exs, loc, w) \Rightarrow seg_2(Exs, loc, w) \notin S$, $\forall seg_1(Exs, loc, w) \in S$.

Thus a segment is the longest common sequence of genomic regions starting at *loc*, such that these regions are spliced similarly, i.e. the entire sequence belongs to the same set of transcripts. That means the three junctions J1, J2, J3 shown in 1 (right) will be concatenated into one segment which captures any read of length L spanning any of these junctions.

2.2 Segmentation Algorithm Overview

Given the transcriptome annotation (GTF format file) and the transcript sequences (FASTA format files) as input, Yanagi generates the set of segments and its sequences (as a FASTA file) as the output of the segmentation process. Figure 2 illustrates an example of how Yanagi perform transcriptome segmentation given the splicing graph of a complex AS event studied in [18]. Recall, that in splicing graphs, nodes represent genomic regions and edges represent how the regions are spliced, while paths represent possible transcripts.

The transcriptome segmentation process can be summarized into three steps: (1) Preprocessing the transcriptome annotation in order to obtain disjoint transcriptome regions, (2) Constructing a Segments Graph (SgG), and finally (3) Generating the segment library. Each transaction in Figure 2 represents one of these three steps.

2.3 Preprocessing

In our algorithm, exons and junctions serve as initial candidates for segment generation. We apply a preprocessing step to eliminate exon overlaps present in the transcriptome reference from events involving alternative 3'/5' splice sites, or transcription start/end sites. This step ensures that any splicing event is occurring either at the beginning or the end of a genomic segment, which makes the process of generating L-disjoint and max covering segments easier. The preprocessing step is independent from the parameter L, so it can be done only once per transcriptome reference. We implemented the preprocessing step based on the GenomicRanges package in R , specifically the *disjoin* function, which takes less than a few seconds to run on the human genome.

2.4 Segments Graph

Currently Yanagi builds a separate segment graph for each gene, since there are no alternative splicing events between transcripts of different genes. However, future work may use segment

Figure 1 An example of naive segments based on exons and junctions. Two cases are shown, each is represented using a splicing graph of two transcripts, along with a set of possible RNA-seq reads and the generated segments following the naive approach. The first case (left) shows a simple case where the naive approach successfully generates segments spanning all possible reads. The second case (right) shows a case of two short exons (E2, E3 of width $k < L$) where the naive approach fails to span the given read.

graphs that connect different genes sharing regions of identical sequence length L or greater, but we have yet to address this.

▶ **Definition 4.** Segment Graph

A segment graph G is an acyclic directed graph defined by the pair (N, E), where N is a set of nodes representing segments, and E is the set of directed edges between the nodes. An edge $e : (n_i, n_j) \in E$ is created if the segment corresponding to node n_i directly precedes the segment corresponding to node n_j in some transcript.

▶ **Definition 5.** Segment Node

A segment node n is a node in segment Graph G that represents an *L-disjoint* and *max-spanning* segment $seg_n(Exs_n, loc_n, w_n)$, such that $w_n \geq L$.

While full details of the algorithm are given in Appendix A, here we present a high-level description. For a given gene, the algorithm iterates over the set of annotated transcripts in that gene. A cursor *loc* starting at the beginning of a transcript slides over the sequence of genomic regions forming that transcript. Given the current cursor location loc_n, a seed of the node n is initiated. Then a refinement step, explained further in the next paragraph, is used to handle cases involving exons shorter than L. The segment node is then added to the graph with the key pair (exs_n, loc_n) as the node identifier, and the cursor *loc* is advanced to the new location. It should be noted that the out-degree of each segment node corresponds to the number of upcoming alternative splices. The final step is generating the actual segments. Any segment with an out-degree greater than one is a candidate start of a segment. Each possible path beginning at a start-segment node till the following start-segment node (or a leaf node) produces an output segment. See Figure 2 and Appendix A for the full details of the segmentation algorithm. It is worth mentioning that a segment graph may look like a de Bruijn graph (DBG) that is commonly used in assembly problems. However, a path of nodes in DBG represents a sequence of k-mer components while a path of nodes in SgG represents a sequence of genomic regions spliced into an isoform. As a result, the DBG built from the list of transcripts and the DBG built from the list of segments should be identical, since both graphs represent the same sequences of nucleotides.

Back to the issue of short regions which raises the possibilities of generating segment nodes of length L spanning more than two exonic regions. Once the key pair (exs_n, loc_n) is determined, the node refinement step determines the extent of that node and how the cursor *loc* should be advanced in order to preserve the L-disjointness constraint. Figure 3

shows a diagram of how a segment node is refined. The logic behind the refinement step is aggregating the sequence of nodes expected to be created spanning the same set of regions exs_n; since it is guaranteed that there are no splicing events occurring between the start and end of region Ex_n, the next necessary segment node would be the node spanning part of Ex_{n+1}. Consequently, the new location of the cursor loc would be the location where the first segment spanning Ex_{n+1} starts. That aggregation improves the time and space complexity of the algorithm as it reduces the number of generated nodes, and avoids shredding the genome in dense areas of short exons which may impose a problem in the quantification step as discussed in next subsection. In fact, the refinement step ensures that for every distinct value of exs, there is a maximum of two segment nodes generated and that reduces the algorithm complexity by factor of L.

As an attempt to analyze the complexity of the algorithm, we can estimate a loose upper bound of the number of segment nodes N in G. Consider a gene with T_g transcripts and E_g disjoint genomic regions (obtained by the preprocessing step), where the maximum width of such a region is w_{max}. Recalling the property mentioned in the previous paragraph, that a maximum of two nodes can be generated for the same set of regions, a segmentation iteration for a transcript that spans $E_t \leq E_g$ regions can generate $1 < o(E_t - \left\lceil \frac{L}{w_{max}} \right\rceil) < o(E_t)$ segment nodes. That gives the upper bound of $o[\sum_{T_g}(E_t - \left\lceil \frac{L}{w_{max}} \right\rceil)]$ or $o[T_g.(E_g - \left\lceil \frac{L}{w_{max}} \right\rceil)]$ for N. That means the time and space complexity of the graph construction increases when using lower values of parameter L, or with organisms of longer and more complex transcriptome structure. Table 1 shows time and memory analysis for constructing the segments library for two organisms used in our later analysis. The results shows that running Yanagi is an efficient and fast process that does not add a burden in terms of time and space requirements.

2.5 Quantification Analysis

After the transcriptome segmentation stage, Yanagi provides the set of generated segments in FASTA format. The segment sequences are accompanied by headers specifying metadata of how each segment was formed, including: gene ID, the set of exonic regions exs included in the segment, start and end locations in the first and last spanned regions, and the set of transcripts corresponding to the segment. The quantification stage starts afterwards by supplying the segment sequences to the preferred kmer-based pseudo-alignment tool, e.g. kallisto, sailfish or RapMap. For single end reads we can obtain segment counts from these tools. In the paired end case, we obtain pseudo alignments from either kallisto or RapMap as a BAM file for each read in the read pair independently. The two generated BAM files are then processed together to obtain segment-pair read counts. A read r_1 is counted toward a pair of segments $< seg_1, seg_2 >$ if the first end is mapped to seg_1 and the second end to seg_2 while both seg_1 and seg_2 have at least one transcript in common. This latter condition ensures that the counts reflects only annotated transcripts, although this condition can be relaxed in principle in favor of identifying unannotated transcripts.

After the segment quantification stage, a count tables for each sample is prepared to be used in the downstream analysis. In this paper we illustrate a workflow based on Yanagi using the problem of differential analysis of exon skipping events across samples from two conditions of interest. By providing the list of annotated splicing events, Yanagi maps each exon skipping event with its corresponding set of segments and sums their counts. For example, an exon skipping event is defined by three exons as in Figure 1 (left). Two segment-level counts are calculated: one from segments spanning the inclusion junction and another from the segments spanning the skipping junction. Note that although segments are

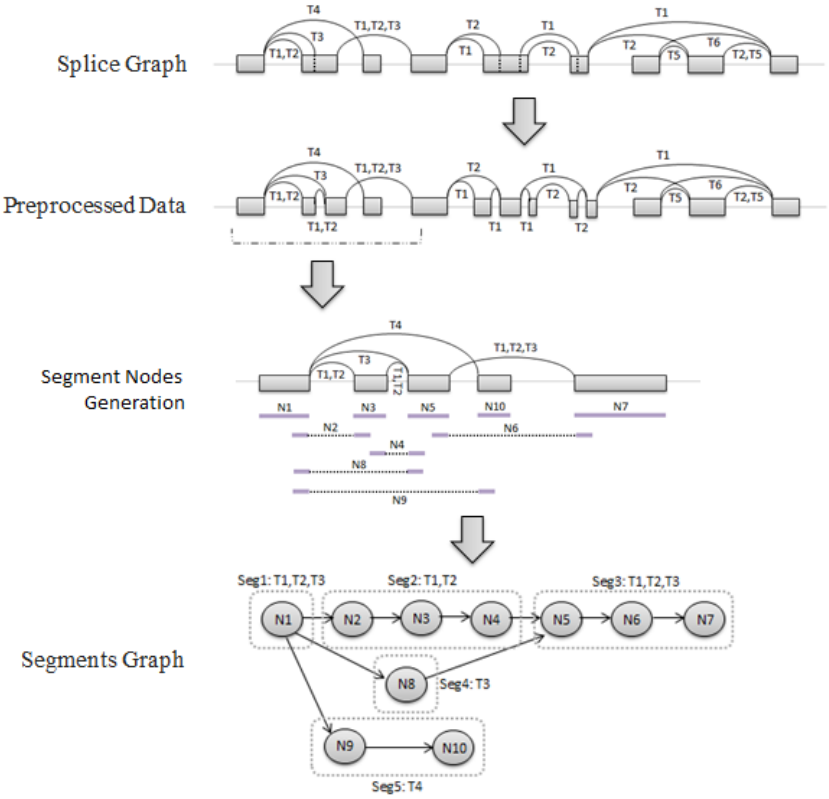

Figure 2 The process of generating segments using the splice graph for an example of a complex splicing event. Each transition represents one of the three main steps of the transcriptome segmentation process. Assuming no short exons for simplicity. Step two and three are cropped to include only the beginning portion of the graph for brevity.

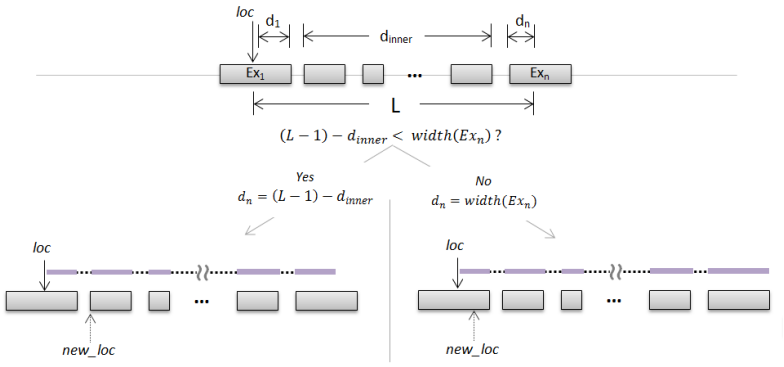

Figure 3 Diagram illustrates the node refinement step, for a node spanning n genomic regions. The step determines the extent of the node and how the cursor *loc* should advance in each of the two candidate cases.

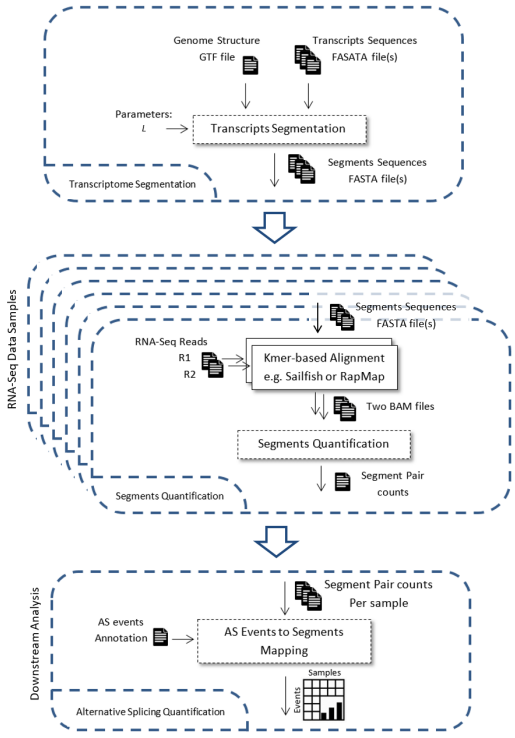

Figure 4 Yanagi-based workflow for alternative splicing analysis, based on paired-end RNA-Seq reads. Dotted blocks are components introduced to assist Yanagi

L-disjoint, if the skipped exon is shorter than L we can have more than one segment spanning the inclusion junctions. Figure 4 illustrates a full workflow based on Yanagi, assuming paired-end reads and targeting AS quantification.

3 Experiments

3.1 Segment Analysis

To analyze the outcome of the segmentation stage, we used Yanagi to build segment libraries for the fruit fly and human genomes: Drosophila melanogaster (UCSC dm6) and Homo sapiens (UCSC hg38) genome assemblies and annotations respectively. These organisms show different genome characteristics, *e.g.* the fruit fly genome has longer exons and transcripts than the human genome, while the number of transcripts per gene is much higher for human genome than the fruit fly. A summary of the properties of each genome is found in [12].

Since L is the only parameter value required by the segmentation algorithm, we tried different values of L to understand the impact of that choice on the generated segments library. Recall that the choice of L is based on the expected read length of the sequencing experiment. For this analysis we chose the set $L = (40, 108, 1000, 10000)$.

Figure 5 shows the histogram of the lengths of the generated segments compared to the the full lengths of the transcripts, for each value of L, for both fruit fly (left) and human (right) genomes. It should be noted that the generated segments should be of at least length L. However, there are the exceptions of segments hitting the end of the transcript where the remaining portion of transcript is shorter than L. The figure shows the expected behavior

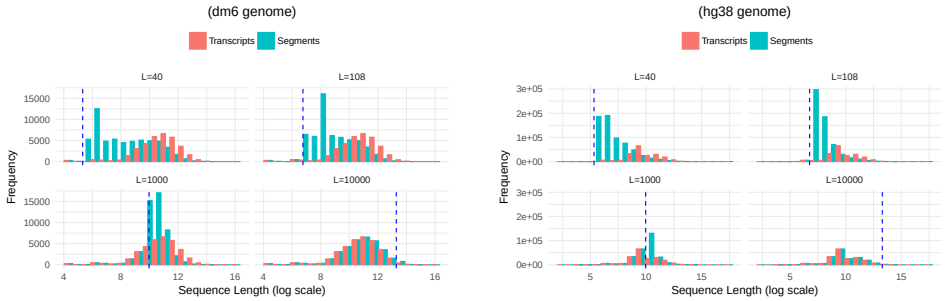

Figure 5 Histogram of transcripts lengths vs. segments lengths for both fruit fly (left) and human (right) genomes, with different values of L (40, 108, 1000, 10,000). Dotted vertical line represents the used value of L during the transcriptome segmentation.

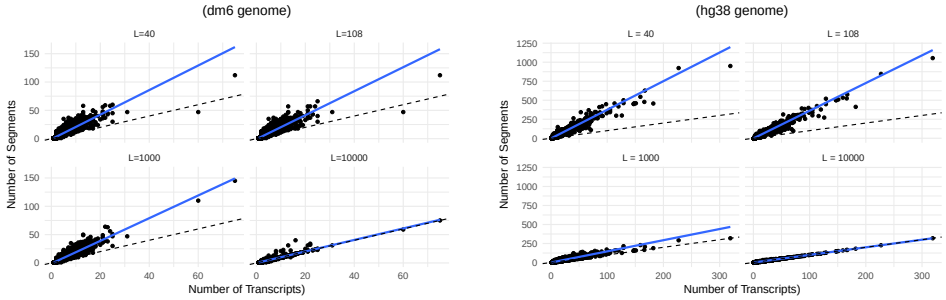

Figure 6 Number of transcripts vs. number of segments, per gene, for both fruit fly (left) and human (right) genomes, with different values of L (40, 108, 1000, 10,000). The figure shows how a fitted line (solid blue) compares to the identity line (dotted black).

when increasing the value of L; using small values of L tends to shred the transcriptome more (higher frequencies for small sequence lengths), especially with genomes of complex splicing structure like the human genome. While with high values of L, such as $L = 10,000$, the minimum segment length required tends to be higher than the length of most transcripts, ending up generating segments such that each segment represents a whole transcript.

Figure 6 shows how the number of generated segments in a gene is compared to the number of the transcripts in that gene, for each value of L, for both fruit fly (left) and human (right) genomes. A similar behavior is observed while increasing the value L, as with the segments length distribution. The fitted line included in each scatter plot provides indication of how the number of target sequences grows compared to the original transcriptome. For example when using $L = 108$, which is a suitable value with Illumina reads, the number of target sequences per gene, which will be the target of the subsequent pseudo-alignment steps, almost doubles. It is clear from both figures the effect of the third step in the segmentation stage. It is important not to shred the transcriptome so much that the target sequences become very short leading to resulting complications in the pseudo-alignment and quantification steps, and not to increase the number of target sequences leading to increasing the processing complexity of these steps.

3.2 Use Case: Differential Exon Skipping

We use the analysis of differential exon skipping events across samples from two conditions of interest as a use case of how to apply segment-level quantification in downstream analysis.

Table 1 Running time (seconds) and memory usage (gigabytes) by Yanagi to generate segment library for fruit fly (Dm6) and human (Hg38) genomes, for both the preprocessing and segmentation steps. Time for the preprocessing step does not include the time to load the FASTA and GTF files. Most of the memory usage is from loading the input data in both steps. Running on a 6-core 2.1 GHz AMD processor, using single-threaded processes. The lower half shows the time and memory usage for running Rapmap's quasi-mapping using the segments library and the the full transcriptome, to quantify samples of 40M paired-end reads, each of length 101bp.

	Dm6		Hg38	
	time(s)	memory(GB)	time(s)	memory(GB)
Preprocessing	13	0.9	112	1.5
Segmentation				
L=40	20	0.4	248	1.3
L=108	20	0.4	250	1.3
L=1000	20	0.4	228	1.3
L=10000	8.5	0.4	77	1.3
Rapmap Indexing (4 Threads)				
L=108	103	0.8	420	2.6
Txs	121	1.1	480	3.7
Rapmap Quantification (8 Threads)				
L=108	236	0.7	220	2.1
Txs	292	1.2	416	3.1

Datasets. The experiments are based on the simulation data provided by [12] for both fruit fly and human organisms (dm3 and hg37 assembly versions, respectively). Each dataset consists of samples from two conditions. Each condition has three replicates. The reads for the replicates are simulated from real RNA-seq samples, to get realistic expression values, after incorporating a variance model and the change required between conditions. The simulation is restricted to only protein-coding genes in the primary genome assembly. The difference in transcripts usage across conditions was simulated in 1000 genes randomly selected from genes with at least two transcripts and high enough expression levels. For each of these 1000 genes, the expression levels of the two most abundant transcripts is switched across conditions. Refer to [12] for full details of the preparation procedure of the dataset.

Differential Splicing Model. Recall that the outcome of the alternative splicing quantification workflow (as in figure 4) is two counts per exon skipping event for each sample: the inclusion count and the exclusion count. The count matrix is then used in a linear model for differential splicing detection for count y_{ij} of segment i, sample j:

$$\log_2 y_{ij} = b_0 + \delta_c(j) \times (2\delta_t(i) - 1) \times b_1 + \delta_t(i) \times b_2$$

with

$$\delta_c(j) = \begin{cases} 1, & \text{if } j \text{ is } \textit{case} \\ 0, & \text{o.w.} \end{cases}$$

$$\delta_t(i) = \begin{cases} 1, & \text{if } i \text{ is } \textit{inclusion} \\ 0, & \text{o.w.} \end{cases}$$

Parameter b_0 models the expected log counts for exclusion segment in control samples, and parameter b_2 models expected log counts for the inclusion segment in control samples. With this, parameter b_1 corresponds to the log-odds ratio for the Percent Spliced In (PSI) statistic frequently used for alternative splicing analysis. Denoting Ψ_0 and Ψ_1 as the PSI values for case and control respectively, $b_1 = 1/2 \log_2 (\frac{\Psi_1}{1-\Psi_1} / \frac{\Psi_0}{1-\Psi_0})$. This model is similar to the DEXSeq model [1], modified to accomodate inclusion/exclusion counts at the single event level.

Preliminary Results. We tested this model over the synthetically generated data for both genomes (using the Limma-voom R Package [11, 7]) using the segments library generated by Yanagi with different values of the parameter L. The preliminary results shown here considers exon skipping events that involve only one inclusion transcript and one exclusion transcript, just to obtain a reasonable level of stability. In addition, only transcripts of high enough expression levels are considered, i.e. the two transcripts have a combined TPM value of 1. Similar constraints is advised in previous analysis [12, 14].

Figure 7 shows ROC plots for sensitivity and specificity measures, for each genome. As a reference, we applied the same linear model on the transcripts' true TPMs provided while preparing the dataset. In principle, the prediction based on the true transcript expression levels (TPMs) would give an upper bound performance, given the particular setting of this synthetic dataset and the provided linear model. Figure 7 shows that using our segments library with the proposed workflow, and using the suitable value of L, gives promising results in detecting differential splicing events (Table 2 shows the AUC values). A suitable value of L would be a value corresponds to the read length of the data, as discussed earlier in 2.1. While using lower value of L intensely shreds the reference into segments shorter than the reads which hurts the quantification step, using higher values rather increases the portion of unnecessary overlap between the generated segments, leading to higher rates of multi-mapped reads.

The results also shows that the segmentation approach gives slightly better performance in the fruit fly case. That matches the fact that the fly transcriptome structure is much simpler than the human transcriptome. Besides, the sequences in the fruit fly are more unique with no allele repeats as the case with the human genome. That makes the counts in the quantification step more stable for the differential analysis.

Table 2 AUC values for the ROC curves in Figure 7 for both fruit fly and human genomes. Including the AUC value when using the transcripts' true TPM, besides AUC values from applying the linear model to the generated segments counts using different values of L.

	Tx.TPM	Seg.Counts			
		$L = 40$	$L = 108$	$L = 1000$	$L = 10000$
Dm3	0.912	0.762	0.889	0.824	0.760
Hg37	0.916	0.706	0.881	0.754	0.727

4 Discussion and Conclusion

In this paper we introduce Yanagi, an efficient tool that creates disjoint segments of reference transcriptomes amenable for quantification of RNA-seq reads using pseudo-alignment techniques. We have formalized the notion of transcriptome segmentation, and proposed an efficient algorithm for constructing L-disjoint, max-spanning segments. We report on the

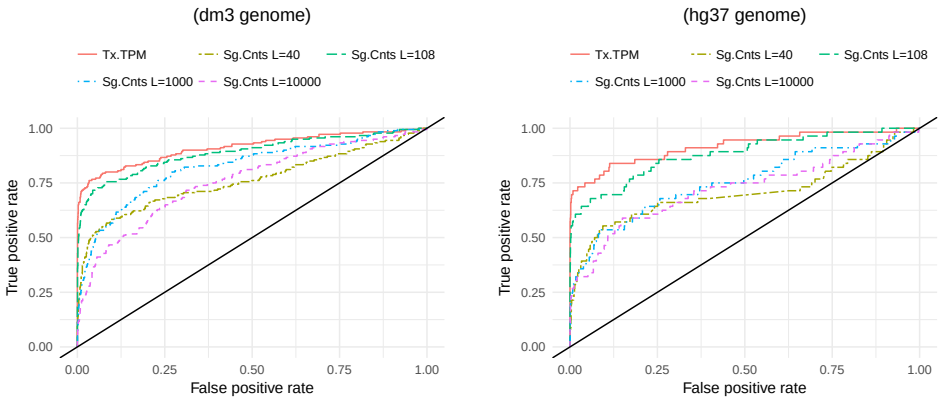

Figure 7 ROC plots for differential alternative splicing (exon skipping events), for both fruit fly (left) and human (right) genomes. ROC curves are included for: transcript-level quantification, besides segment-level quantification using different values of the parameter L.

characteristics of segment libraries in Drosophila melanogaster and Homo sapiens, and use the resulting segments in a use case of differential analysis of exon skipping events across samples in two conditions of interest.

Although it may appear that the discussed Yanagi-based workflow can perform quantification only for the annotated transcripts, the workflow can be extended to discover unannotated transcripts. An unannotated junction can be detected during the segment quantification stage by relaxing the restriction of accepting alignments of a pair of reads only if the pair of segments belong to at least one annotated transcript. When this restriction is relaxed, an unannotated junction can be detected when reads show enough evidence of that junction. I.e. when a segment pair that has no transcripts in common has high enough count.

Finally, the issues of paralogs and intersecting genes are not tackled in the scope of this paper. However, it is clear that there is no extra alignment complexity added due to these issues over the transcriptme-based alignment. Consequently, the occurrence of multi-mapping resulting from such cases also remains the same as the transcriptome-based quantification. So a warranted extension to the current approach of Yanagi is to consider distinct genes that share identical exonic regions of length greater than L altogether.

The concept of transcriptome segmentation, and a tool that can build a segments library, opens the door for more extended analysis than just the use case mentioned in this paper. For instance, segment counts can serve as statistics into algorithms for differential isoform usage analysis, for which existing pseudo-alignment methods are commonly used. Moreover, segment level quantification can provide much more flexible opportunities for analysis including quantification of RNA editing or other non splicing variations. Currently we are exploring the possibility of utilizing the concept of segmentation into the problem of variant calling.

Acknowledgements. We want to thank Julien Buchbinder for preliminary work in transcript segmentation. This work was partially supported by NSF grant ABI 1564785 to SMM and MKG, and NIH grants HG005220 and GM114267 to HCB, MKG and SC.

References

1 Simon Anders, Alejandro Reyes, and Wolfgang Huber. Detecting differential usage of exons from rna-seq data. *Genome research*, 22(10):2008–2017, 2012.

2 Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic rna-seq quantification. *Nature biotechnology*, 34(5):525–527, 2016.

3 Brian J Haas, Arthur L Delcher, Stephen M Mount, Jennifer R Wortman, Roger K Smith Jr, Linda I Hannick, Rama Maiti, Catherine M Ronning, Douglas B Rusch, Christopher D Town, et al. Improving the arabidopsis genome annotation using maximal transcript alignment assemblies. *Nucleic acids research*, 31(19):5654–5666, 2003.

4 Steffen Heber, Max Alekseyev, Sing-Hoi Sze, Haixu Tang, and Pavel A. Pevzner. Splicing graphs and est assembly problem. *Bioinformatics*, 18(suppl_1):S181, 2002.

5 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(4):R36, 2013.

6 Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.

7 Charity W Law, Yunshun Chen, Wei Shi, and Gordon K Smyth. Voom: precision weights unlock linear model analysis tools for rna-seq read counts. *Genome biology*, 15(2):R29, 2014.

8 Bo Li and Colin N Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC bioinformatics*, 12(1):323, 2011.

9 Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 2017.

10 Rob Patro, Stephen M. Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462–464, May 2014.

11 Gordon K Smyth et al. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Stat Appl Genet Mol Biol*, 3(1):3, 2004.

12 Charlotte Soneson, Katarina L Matthes, Malgorzata Nowicka, Charity W Law, and Mark D Robinson. Isoform prefiltering improves performance of count-based methods for analysis of differential transcript usage. *Genome biology*, 17(1):12, 2016.

13 Avi Srivastava, Hirak Sarkar, Nitish Gupta, and Rob Patro. Rapmap: a rapid, sensitive and accurate tool for mapping rna-seq reads to transcriptomes. *Bioinformatics*, 32(12):i192, 2016.

14 Mingxiang Teng, Michael I Love, Carrie A Davis, Sarah Djebali, Alexander Dobin, Brenton R Graveley, Sheng Li, Christopher E Mason, Sara Olson, Dmitri Pervouchine, et al. A benchmark for rna-seq quantification pipelines. *Genome biology*, 17(1):74, 2016.

15 Hagen Tilgner, Fereshteh Jahanbani, Tim Blauwkamp, Ali Moshrefi, Erich Jaeger, Feng Chen, Itamar Harel, Carlos D Bustamante, Morten Rasmussen, and Michael P Snyder. Comprehensive transcriptome analysis using synthetic long-read sequencing reveals molecular co-association of distant splicing events. *Nature biotechnology*, 33(7):736–742, 2015.

16 Cole Trapnell, Lior Pachter, and Steven L Salzberg. Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–1111, 2009.

17 Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J Van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature biotechnology*, 28(5):511–515, 2010.

18 Jorge Vaquero-Garcia, Alejandro Barrera, Matthew R. Gazzara, Juan Gonzalez-Vallinas, Nicholas F. Lahens, John B. Hogenesch, Kristen W. Lynch, Yoseph Barash, and Juan

Valcárcel. A new view of transcriptome complexity and regulation through the lens of local splicing variations. *eLife*, 5:e11752+, February 2016.

19 S Lawrence Zipursky, Woj M Wojtowicz, and Daisuke Hattori. Got diversity? wiring the fly brain with dscam. *Trends in biochemical sciences*, 31(10):581–588, 2006.

A Transcriptome Segmentation Algorithm

Algorithm 1 Yanagi's Segments Library Generation

Require: Transcriptome Annotation (GTF File), Transcripts Sequences (FASTA Files)

$TxDB \leftarrow makeTxDBFromGTFFile$ ▷ Preprocessing Step
$DExs \leftarrow disjoinExons(TxDB)$
$TxDB \leftarrow adjustTxDB(TxDB, DExs)$

Step 2 - Segment Graph Construction

procedure CONSTRUCT_SEG_GRAPH($TxDB, g, L$) ▷ SgG of gene g
  $G \leftarrow emptygraph$
  $St \leftarrow \phi$
  $prev \leftarrow$ DUMMY_NODE
  for each $Tx \in \mathcal{T}xDB(g)$ **do** ▷ For each transcript in gene g
    $loc \leftarrow start(Tx)$
    while $loc < end(Tx)$ **do** ▷ Iterate till the end of transcript
      $gr \leftarrow GenomicRange(Tx, loc, loc + L)$
      $Exs \leftarrow exons[TxDB(gr)]$
      $w, loc_{new} \leftarrow$ REFINE_NODE(Exs, loc, L) ▷ Node refinement step
      $node \leftarrow getOrCreateNode(G, < Exs, loc, w >)$
      $Txs(node) \leftarrow Txs(node) + Tx$
      $Next(prev) \leftarrow Next(prev) + node$ ▷ Make an edge
      if $Txs(prev) \neq Txs(node)$ **then** ▷ Mark branches in graph
        for each $n \in Next(prev)$ **do**
          $St \leftarrow St + n$
      $prev \leftarrow node$ ▷ Advance loop
      $loc \leftarrow loc_{new}$
  return G, St ▷ The SgG of gene g

Step 3 - Segments Library Generation

procedure GENERATE_SEGMENTS(G, St)
  for each $node \in St$ **do** ▷ Iterate over branching points in G
    $seg \leftarrow newsegment$ ▷ Initialize a new segment
    $seg.appendNode(node)$
    while $|Next(node)| = 1$ **do** ▷ Aggregating chain of nodes into the segment
      $node \leftarrow Next(node)$
      $seg.appendNode(node)$
    $outputSegment(seg)$