Dissertation zur Erlangung des Doktorgrades
der Fakultät für Chemie und Pharmazie
der Ludwig–Maximilians–Universität München

Computational methods for RNA splicing

Isra'a Ahmad Fawzi Al-Qassem

aus

Irbid, Jordan

2021

Erklärung:
Diese Dissertation wurde im Sinne von §7 der Promotionsordnung vom 28. November 2011 von Herrn Dr. Stefan Canzar betreut.

Eidesstattliche Versicherung:
Diese Dissertation wurde eigenständig und ohne unerlaubte Hilfe erarbeitet.

München, 29 October 2021

Isra'a Ahmad Fawzi Al-Qassem

Dissertation eingereicht am 1 October 2021
1. Gutachter: Dr. Stefan Canzar
2. Gutachter: Prof. Dr. Julien Gagneur
Mündliche Prüfung am 25 October 2021

Contents

Acknowledgments

I would like to express my sincere gratitude to my advisor Dr. Stefan Canzar for his invaluable advice, continuous guidance, and patience during my PhD study. Vielen dank, Stefan. I would like to thank the following people for helping with the McSplicer project, Heejung Shim, assistant professor at the Melbourne Integrative Genomics, thanks Heejung for all your help, insightful comments and suggestions. My appreciation also goes to my friends and collaborators, Yash Sonthalia and Erika Klitzke-Feser.

Special thanks to Domagoj Matijevic, associate professor of computer science at the University of Osijek, and his student Antonio Jovanovic for their help with the RNA Ptr-Nets project. I also would like to thank Prof. Dr. Johannes Stigler for sharing his lab GPU machine with us.

My gratitude extends to the Graduate school of Quantitative Biosciences Munich (QBM) for funding my PhD studies during the first two years. I also would like to thank previous and current QBM staff, especially Mirely Kollmannsberger, Dr. Markus Hohle, and Dr. Dietmar Martin for their valuable academic and administrative advice.

Last but not least, I would like to express my gratitude to my family and friends in Munich and all around the world who went through this journey with me, always offering love and support.

Summary

In this thesis, we develop two computational methods to tackle two different RNA splicing challenges. In Chapter 2, we present McSplicer, a probabilistic model for estimating splice site usage from RNA-seq data. The main contribution of McSplicer is to quantify alternative splicing using a simplified probabilistic model of the underlying splicing process, instead of quantifying local splicing events or full-length transcripts. McSplicer is based on the usage of individual splice sites and can estimate arbitrarily complex types of alternative splicing patterns. In various experiments, McSplicer demonstrates more accurate estimates compared to other competing methods in literature.

In Chapter 3, we introduce RNA Ptr-Nets, a deep learning approach for predicting RNA splicing branchpoints. Alternative branchpoint selection plays a role in alternative splicing and diseases, nonetheless human branchpoint annotations are still incomplete and only a small fraction of branchpoints is experimentally verified. RNA Ptr-Nets aims to predict all branchpoints within an intronic region to overcome the limitation of existing machine learning and deep learning approaches which either require genome annotation and a great amount of feature engineering or predict only a single brachpoint site associated with each acceptor site.

CHAPTER 1

Introduction

During RNA splicing, precursor messenger RNA (pre-mRNA) transcript is transformed into a mature messenger RNA (mRNA). Introns (i.e., non-coding regions) are omitted while exons (i.e., coding regions) remain in the mRNA. Splicing is an essential process in eukaryotic cell development and has been linked to various diseases. Here, we propose two computational methods to tackle two different RNA splicing challenges.

First, we propose McSplicer, a probabilistic model for detecting alternative splicing. Alternative splicing removes intronic sequences from pre-mRNAs in alternative ways to produce different forms (isoforms) of mature mRNA. The composition of expressed transcripts gives specific functionalities to cells in a particular condition or developmental stage. In addition, a large fraction of human disease mutations affect splicing and lead to aberrant mRNA and protein products. Current methods that interrogate the transcriptome leverage RNA sequencing technology (RNA-seq). RNA-seq produces short reads from which existing methods infer and quantify RNA splicing. Existing alternative splicing quantification approaches either suffer from short-read length when trying to infer full-length transcripts, or are restricted to predefined units of alternative splicing that they quantify from local read evidence.

Instead of attempting to quantify individual outcomes of the splicing process such as local splicing events or full-length transcripts, we propose to quantify alternative splicing using a simplified probabilistic model of the underlying splicing process. Our model is based on the usage of individual splice sites and can generate arbitrarily complex types of splicing patterns. In our implementation, McSplicer, we estimate the parameters of our model using all read data at once and we demonstrate in our experiments that this yields more accurate estimates compared to competing methods. Our model is able to describe multiple effects of splicing mutations using few, easy to interpret parameters, as we illustrate in an experiment on RNA-seq data from autism spectrum disorder patients.

Second, we propose RNA Ptr-Nets, a deep learning approach for predicting RNA splicing branchpoints. RNA splicing requires three main signals i.e., the donor splice site (5'SS), the acceptor splice site (3'SS), and the branchpoint (BP) site. These three signals work together in a two-step mechanism of RNA splicing. First, the pre-mRNA is cut at the donor site where the 5' end of the intron is joined to the BP site and generates a lariat intermediate molecule. Then, the last nucleotide of the intron at the 3' end is cut, and the two exons are joined together while the

intron lariat is degenerated quickly (see Fig. 1.1). BPs are mandatory signals in RNA splicing, equally as important as acceptor and donor sites. Moreover, previous research shows that alternative BP selection plays a role in alternative splicing and diseases (Corvelo et al., 2010; Alsafadi et al., 2016).

In theory, the position of BP can be determined using RNA-seq reads spanning the donor site and BP junctions, however the intron lariat is degenerated quickly during RNA splicing which makes such reads less frequent (Taggart et al., 2017). To date, human BP annotations are still incomplete and only a small fraction of BPs is experimentally verified.

Over the past decade, many machine learning approaches have emerged to provide genome-wide annotation of BPs in human gene introns using gene annotations and genomic sequence. Those approaches either require genome annotation and a decent amount of feature engineering prior to the training step or predict only a single BP site for each acceptor site (i.e., cannot annotate all BPs within an intron).

Here, we introduce RNA Pointer Networks (RNA Ptr-Nets) based on the novel deep learning architecture, Pointer Networks (Ptr-Nets) (Vinyals et al., 2015). RNA Ptr-Nets model aims to predict all BPs associated with each acceptor site. Our model takes as input intronic sequences and outputs pointers to BP positions with respect to the input sequence, further it does not require genome annotation nor any feature engineering.

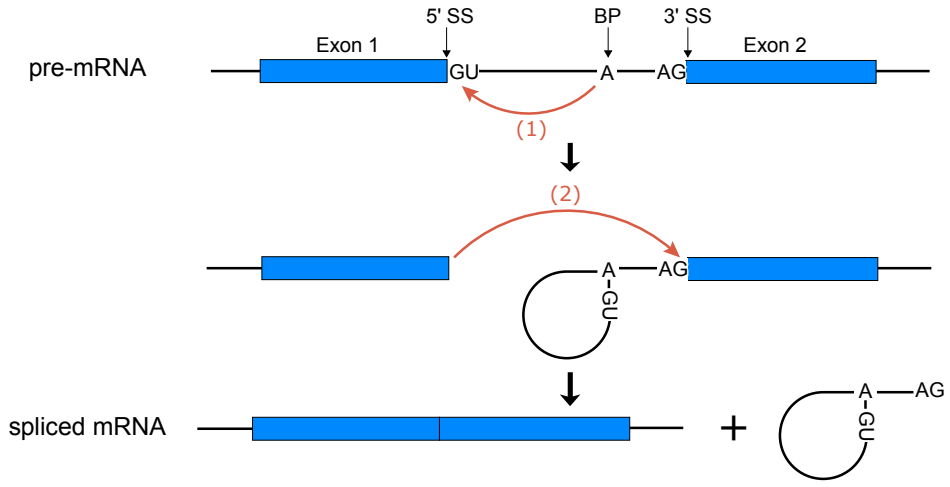


Figure 1.1: The two-step mechanism of RNA splicing. First, the pre-mRNA is cut at the 5'SS where the 5' end of the intron is joined to the BP site and generates a lariat intermediate molecule. Then, the last nucleotide of the intron at the 3' end is cut, and the two exons are joined together while the intron lariat is degenerated quickly.

CHAPTER 2

McSplicer: a probabilistic model for estimating splice site usage from RNA-seq data

This chapter is adapted with minimal changes from the publication: **Alqassem, Israa**, Yash Sonthalia, Erika Klitzke-Feser, Heejung Shim, and Stefan Canzar. "McSplicer: a probabilistic model for estimating splice site usage from RNA-seq data." Bioinformatics 37, no. 14 (2021): 2004-2011.

Through alternative splicing (AS), a single gene can produce multiple mRNA transcripts, or isoforms, that combine exons in alternative ways. Approximately 95% of human multi-exon protein-coding genes undergo alternative splicing (Pan et al., 2008), creating a remarkably complex set of transcripts that give specific functionalities to cells and tissues in a particular condition or developmental stage.

RNA sequencing (RNA-seq) is routinely used in genome-wide transcript analysis. This technology produces short reads from which existing methods infer and quantify RNA splicing, broadly, in one of two different ways. Methods either analyze full-length transcripts or focus on individual splicing events. Transcript assembly methods such as StringTie (Pertea et al., 2015), CIDANE (Canzar et al., 2016), and CLASS (Song and Florea, 2013) aim to identify the set of expressed full-length transcripts which in principle provides a complete picture of all splicing variations, see e.g. transcript t_1-t_5 in Fig. 2.1. The transcript assembly problem is, however, ill-posed (Lacroix et al., 2008) and error-prone especially for complex genes expressing multiple transcript isoforms (Hayer et al., 2015).

Event-based methods, therefore, focus on local splicing patterns such as the classical exon skipping event denoted in Fig. 2.1, without a prior attempt to assemble or quantify full-length transcripts. The relative abundance of different splicing outcomes that can potentially be shared by multiple transcripts, can then be quantified using a simple metric such as percent spliced in (PSI) (Venables et al., 2008). A notable exception is SUPPA (Alamancos et al., 2015) which derives PSI values from quantified transcript abundances.

Event-based methods differ in the complexity of the units of AS they quantify. In the simplest case, methods such as MISO (Katz et al., 2010), SUPPA, ASGAL (Denti et al., 2018), SpliceGrapher (Rogers et al., 2012), and SplAdder (Kahles et al., 2016) identify one of the canonical types of AS, such as exon skipping, alternative 5' and 3' splice sites, intron retentions, and mutually exclu-

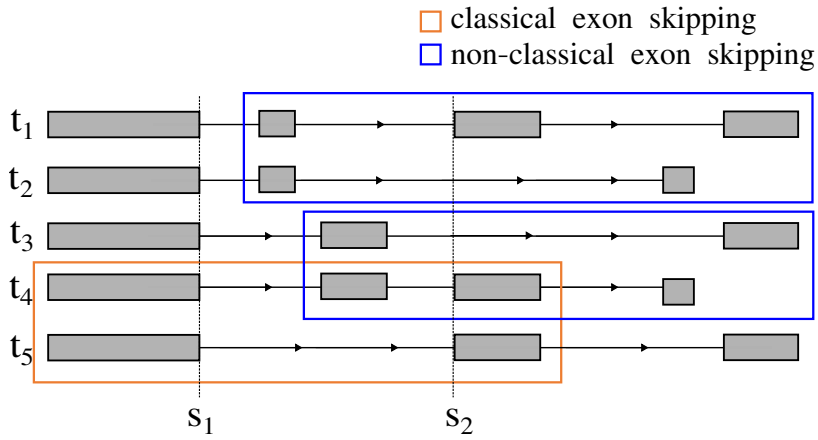


Figure 2.1: Complex alternative splicing involving 5 different transcripts. The two classical exon skipping events between t_1 and t_5, and between t_4 and t_5 do not fully capture the overall complexity. The two exon skippings marked in blue are not considered classical events and would not be reported by methods such as SplAdder, since they also differ in the last exon. Methods such as MAJIQ generalize simple events to more complex AS units that contain all introns sharing a common splice site. Two such AS units are required to describe the simple exon skipping event marked in orange, one comprising three introns sharing donor s_1 and one containing three different introns sharing acceptor s_2.

sive exons (see Fig. 2.2). In Fig. 2.1, this definition would include the two simple exon skippings between t_1 and t_5 and between t_4 and t_5, and mutually exclusive spliced exons in t_1 and t_4, clearly underestimating the full AS complexity across t_1-t_5.

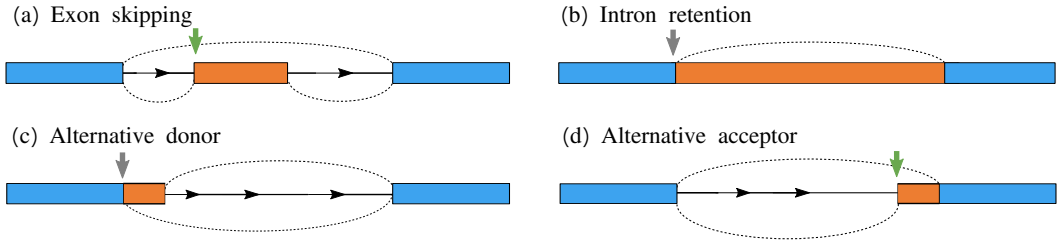


Figure 2.2: Four types of simple alternative splicing events. Blue rectangles represent constitutive exon or exonic segments. Orange rectangles represent alternatively spliced ones. (a) The usage of the marked acceptor site defines the relative abundance of the inclusion of the skipped exon. (b) For intron retentions, the usage of the marked splice site defines the relative abundance of the inclusion of the intron. For alternative donors (c) and alternative acceptors (d), the usage of the marked donor and acceptor sites determine the relative abundance of the two alternative events.

Compared to these simple types of splicing events, complex events involve multiple alternative splice sites or exons and according to Vaquero-Garcia et al. (2016) constitute at least one-third of AS events observed in human and mouse tissues. Methods such as JUM (Wang and Rio, 2018), MAJIQ (Vaquero-Garcia et al., 2016) and the method proposed in Oesterreich et al. (2016), therefore consider AS units that generalize simple events to more complex patterns. They quantify the relative usage of an arbitrary number of introns that share a common splice site. Since these AS units capture only the common endpoints of alternative splicing patterns, such methods need to quantify two AS units for a single exon skipping event (Fig. 2.1). LeafCutter (Li et al., 2018) and

Whippet (Sterne-Weiler et al., 2017) add further introns to AS units. At the extreme end, Whippet enumerates all possible transcript fragments that combine overlapping events and estimates their relative abundance using an EM algorithm similar to full-length transcript quantification methods such as kallisto (Bray et al., 2016).

PSGInfer (LeGault and Dewey, 2013) quantifies alternative splicing based on Probabilistic Splice Graphs (PSGs). It assigns weights to the edges of a splicing graph (Heber et al., 2002) using parameters that describe the splicing process, rather than focusing on individual outcomes of the splicing processes such as local splicing events or full-length transcripts. The parameter estimates can then be used to estimate transcript and processing event frequencies. Motivated by the work by LeGault and Dewey (2013), we similarly aim to quantify alternative splicing by building a probabilistic model as a simple approximation to the underlying splicing processes. In constrast to PSG, however, our model employs the usages of annotated as well as novel splice sites across all expressed transcripts to describe a simplified splicing process that has generated the set of expressed transcripts. Traversing the linear ordering of all exons of a gene from 5' to 3', the usage of each splice site specifies the probability with which the site is used as donor or acceptor site. For example, the usage of acceptor s_2 in Fig. 2.1 indicates the abundance of transcripts t_1, t_4, and t_5 that "use" the acceptor relative to the total output t_1-t_5 of the gene. Our model assumes that splice site usages are independent of each other, which allows for a computationally more efficient estimation of parameters compared to PSGInfer.

This model by definition can generate complex splicing patterns that do not rely on any predefined simple or complex AS units as event-based methods like SplAdder, MAJIQ, or LeafCutter do. At the same time, splice site usages that capture simultaneous changes in multiple isoforms facilitate the interpretation of point mutations that disrupt splicing as is the case in many genetic disorders (Anna and Monika, 2018). Instead of attempting to quantify each one of multiple possible effects on intron or even transcript level, a reduced splice site usage as computed by McSplicer may directly reflect the weakening of a splice site by a point mutation in the consensus splice site sequence that is responsible for these effects, as we illustrate in our experiments on RNA-seq data from autism spectrum disorder patients (Section 2.3.4).

Furthermore, our method simultaneously estimates the model parameters, i.e. splice site usages, using all reads mapped to a gene locus, often resulting in more accurate estimates compared to event-based methods that use only reads directly supporting their parameters. We demonstrate the improved accuracy of McSplicer compared to existing methods in our experiments.

2.1 Method overview

A typical RNA-seq analysis workflow that uses McSplicer to estimate the usage of splice sites consists of the five steps illustrated in Fig. 2.3. After A) mapping reads in an RNA-seq sample to a reference genome sequence using a read alignment tool such as STAR (Dobin et al., 2013) or HISAT (Kim et al., 2015), we B) assemble reads to full-length transcripts using methods such as StringTie (Pertea et al., 2015) or CLASS (Song and Florea, 2013) to identify annotated as well as novel splice sites. Step B) can be omitted and instead a curated catalog of known transcripts may be provided. In both cases, McSplicer does not rely on any transcript-level phasing of exons but uses the extracted splice sites and transcription start (TSS) and end sites (TES) to C) partition a gene into contiguous, non-overlapping segments. Segments are defined as minimal subsequences of a gene's exons and introns that are bounded by splice sites, TSS, or TES. The example shown in Fig. 2.3 C contains 6 such segments. We count reads that overlap distinct combinations of

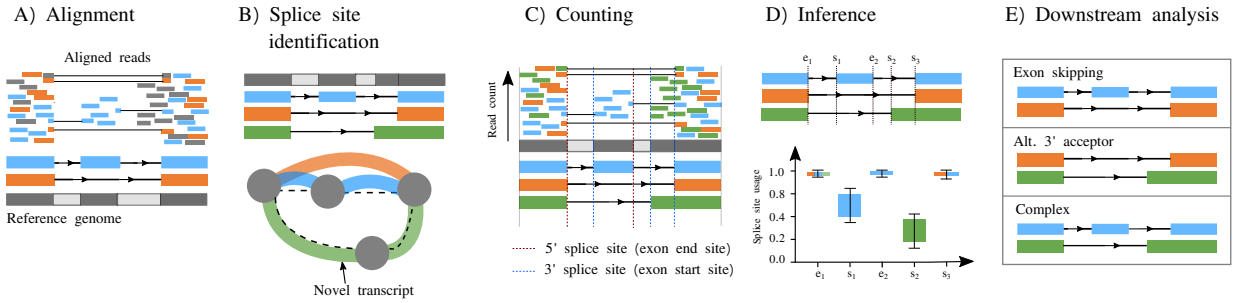
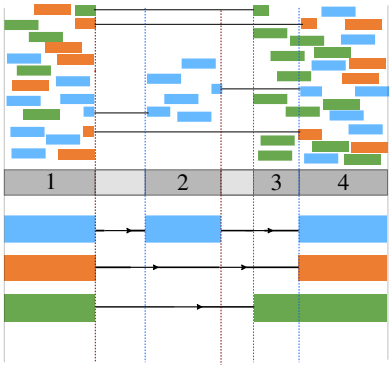


Figure 2.3: McSplicer workflow summary. The main steps of the McSplicer analysis are: A) Map RNA-seq reads to the reference genome sequence. B) Identify annotated as well as novel splice sites through the reference-based assembly of transcripts using, e.g., StringTie (Pertea et al., 2015). C) Divide the gene into non-overlapping segments bounded by splice sites, TSS and TES and count the number of reads mapping to distinct combinations of segments. In this example, only the start of the first exon and the end of the last exon are bounded by TSS and TES, respectively, the remaining exon start and end sites correspond to splice sites. D) Estimate splice site usages using McSplicer. E) Leverage splice site usages in various kinds of downstream analyses, such as the quantification of different types of alternative splicing events.

such segments. The precise sequence of segments a mapped read overlaps defines its mapping *signature* (Canzar et al., 2016). Reads that map to the same signature are equivalent in terms of the splicing pattern they represent. From *signature counts*, i.e. the number of reads mapping to the same signature (see Fig. 2.4 for an illustration), McSplicer estimates splice site usages in step (D). Splice site usages computed by McSplicer can be leveraged in E) different types of downstream anlyses, including the quantification of various types of splicing events.



signature	signature count
$\{1\}$	16
$\{1-2\}$	1
$\{1-3\}$	1
$\{1-4\}$	2
$\{1-2-4\}$	0
$\{1-3-4\}$	0
$\{2\}$	4
$\{2-4\}$	1
$\{3\}$	4
$\{3-4\}$	4
$\{4\}$	15

Figure 2.4: An illustrative example showing signatures with their corresponding read counts. McSplicer estimates exon start and end site usages from these *signagture counts* rather than from individual read alignments. In this example, three transcripts imply a partitioning into 6 segments, 4 of which are part of exons and contain reads. Read colors indicate the originating transcript.

In the following sections, we introduce McSplicer's model and algorithm for the estimation of parameters in that model. A more detailed description of the model and algorithms is provided in Section Method details 2.2. In the technical description of our model, we refer to the exon boundaries at the $3'$ (acceptor) splice site and at the TSS as exon start sites, and to the $5'$ (donor) splice sites and TES as exon end sites. The description of our model is based on single-end reads

which we apply to paired-end reads in Section 2.3.3. In the next section, we recapitulate the commonly assumed generative model of RNA-seq that also underlies the McSplicer model. For the sake of simplicity, we introduce the model based on individual observed reads and explain how parameters can be estimated from (much fewer) signature counts at the end of Section 2.1.3.

2.1.1 A generative model for RNA-seq reads

Consider the RNA-seq reads that mapped to a given gene. Reads are derived from one end of each of N fragments and each read has length L. We assume that each fragment is independently generated from one of the possible transcripts allowed by our model (see next Section). In this section, we describe a generative model for the sequence of the n-th read R_n. The probability of R_n can be written as

$$\Pr(R_n) = \sum_t \Pr(R_n|T_n = t)\Pr(T_n = t), \tag{2.1}$$

where T_n represents the transcript from which R_n was generated. Following models in Li et al. (2010) and LeGault and Dewey (2013), we assume that the probability of generating R_n from a transcript t is proportional to the product of the (effective) length of the transcript, $l(t)$, and the relative abundance of the transcript, $w(t)$:

$$\Pr(T_n = t) = \frac{l(t)w(t)}{\sum_{t'} l(t')w(t')}. \tag{2.2}$$

The effective length of a transcript denotes the number of possible start position of a sampled read (Trapnell et al., 2010). We introduce B_n that denotes the start position of R_n in T_n, leading to

$$\Pr(R_n|T_n = t) = \sum_{b=1}^{l(t)} \Pr(R_n|B_n = b, T_n = t)\Pr(B_n = b|T_n = t). \tag{2.3}$$

Making the simplifying assumption that R_n was generated uniformly across transcript t, we have

$$\Pr(B_n = b|T_n = t) = \frac{1}{l(t)}. \tag{2.4}$$

$\Pr(R_n|B_n = b, T_n = t) = 1$ if R_n is identical to the sequence of length L starting at a position b in transcript t, and this probability is 0 otherwise.

2.1.2 McSplicer: an inhomogeneous Markov chain to model the relative abundance of transcripts

We propose a new model for the relative abundance of transcripts expressed by a gene, denoted by $w(t)$ in the previous section. Suppose we have obtained in step (B) in the McSplicer workflow (Fig. 2.3) exon start sites, $s_1, \ldots, s_{M_s}$, and exon end sites, $e_1, \ldots, e_{M_e}$, ordered by their occurrence in forward direction of a given gene. Here, we do not include the start site of the first exon and the end site of the last exon, since the former is treated differently in our model (see below) and the usage of the latter is always equal to 1 in our model. All exon start and end sites partition the gene into non-overlapping segments $X_1, \ldots, X_M$, where $M = M_s + M_e + 1$ and each segment is defined by a region enclosed by splice sites or transcription start or end sites that occur consecutively along the genome (see Fig. 2.3C and Fig. 2.5). We introduce a sequence of hidden variables, $Z = (Z_1, \ldots, Z_M)$, where

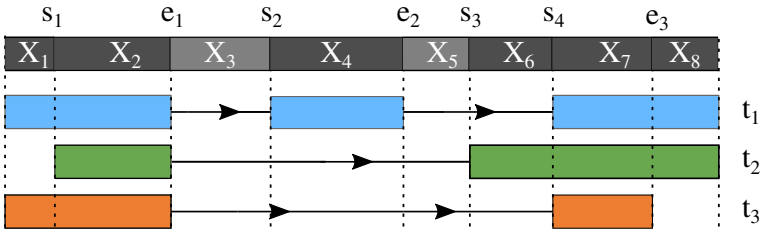


Figure 2.5: Example of hidden states representing 3 different transcripts. Five exon start sites and four exon end sites divide the gene into eight segments. Note, however, that the TSS bounding X_1 from the left and the TES bounding X_8 from the right are not labeled here since our model treats them differently (see main text). Therefore, $M_s = 4$, $M_e = 3$, and $M = 8$. The three sequences of states of Z, $(1,1,0,1,0,0,1,1)$, $(0,1,0,0,0,1,1,1)$, and $(1,1,0,0,0,0,1,0)$, represent the three transcripts t_1, t_2, and t_3, respectively.

Z_i is a binary indicator for whether the i-th segment X_i is transcribed ($Z_i = 1$). Then, a particular transcript can be represented by a sequence of states for Z, as illustrated for transcripts t_1, t_2, t_3 in Fig. 2.5. Thus, we can model the relative abundance of transcripts by modelling the probability of Z.

We use an inhomogeneous Markov chain to model the probability of the sequence of hidden variables, $Z = (Z_1, \ldots, Z_M)$. Specifically, the initial probability is given by

$$\Pr(Z_1 = 1) = \pi, \tag{2.5}$$

where π represents the proportion of transcripts that contain the first segment. We model the transition probability from Z_i to Z_{i+1} for $i = 1, \ldots, M-1$ as follows. If two consecutive segments X_i and X_{i+1} are separated by an exon start site s_m,

$$\Pr(Z_{i+1} = 1 | Z_i = 0) = p_m \tag{2.6}$$
$$\Pr(Z_{i+1} = 1 | Z_i = 1) = 1. \tag{2.7}$$

If they are separated by an exon end site e_m,

$$\Pr(Z_{i+1} = 0 | Z_i = 1) = q_m \tag{2.8}$$
$$\Pr(Z_{i+1} = 0 | Z_i = 0) = 1. \tag{2.9}$$

That is, if the current segment is transcribed ($Z_i = 1$), the splicing process ignores an exon start site (Equation 2.7), but it considers the potential usage of an exon end site e_m and decides to use it, i.e. end the exon, with its usage probability q_m (Equation 2.8). On the other hand, if the current segment is not transcribed ($Z_i = 0$), the splicing process ignores an exon end site (Equation 2.9), but it uses an exon start site s_m with its usage probability p_m (Equation 2.6). The parameters $p = (p_1, \ldots, p_{M_s})$ and $q = (q_1, \ldots, q_{M_e})$ represent probabilities of using the corresponding exon start and end sites, respectively, given that each site is considered for potential usage. Throughout the rest of this work, we refer to these usage probabilities simply as usages. Table 2.1 shows the relative abundances defined by the proposed model for the three transcripts presented in Fig. 2.5. A more detailed description is provided in Sections 2.2.1, 2.2.2 and 2.2.3.

transcript t	$Z = (Z_1, \ldots, Z_8)$	$w(t) = \Pr(Z_1, \ldots, Z_8)$
t_1	$z_{[1:8]}(1,1,0,1,0,0,1,1)$	$\pi \times 1 \times q_1 \times p_2 \times q_2 \times (1-p_3) \times p_4 \times (1-q_3)$
t_2	$z_{[1:8]}(0,1,0,0,0,1,1,1)$	$(1-\pi) \times p_1 \times q_1 \times (1-p_2) \times 1 \times p_3 \times 1 \times (1-q_3)$
t_3	$z_{[1:8]}(1,1,0,0,0,0,1,0)$	$\pi \times 1 \times q_1 \times (1-p_2) \times 1 \times (1-p_3) \times p_4 \times q_3$

Table 2.1: The relative abundances defined by the McSplicer model for the three transcripts presented in Fig. 2.5.

2.1.3 Parameter estimation and uncertainty quantification

We use an EM algorithm to compute the maximum likelihood estimates for the model parameters $\Theta = \{\pi, p, q\}$, that is $\hat{\Theta} := \text{argmax}_\Theta \Pr(R_1, \ldots, R_N|\Theta)$. The complete log likelihood in the EM algorithm involves $\Pr(R_n, B_n = b, T_n = Z|\Theta)$ for $b \in \{1, \ldots, l(Z)\}$ (Section 2.2.4). By combining the generative model and the McSplicer model in the previous two sections, $\Pr(R_n, B_n = b, T_n = Z|\Theta)$ can be written as

$$\begin{aligned} &\Pr(R_n|B_n = b, T_n = Z) \Pr(B_n = b|T_n = Z) \Pr(T_n = Z|\Theta) \\ &= \frac{1}{l(Z)} \frac{l(Z) w_\Theta(Z)}{\sum_{Z'} l(Z') w_\Theta(Z')} = \frac{w_\Theta(Z)}{\sum_{Z'} l(Z') w_\Theta(Z')} \end{aligned} \tag{2.10}$$

if R_n is identical to the sequence of length L starting at position b in transcript Z. Otherwise, this probability is 0. The details of the application of the EM algorithm to the proposed model are provided in Section 2.2.4. The EM algorithm uses several quantities that we compute using dynamic programming, see Section 2.2.5. Also, all quantities required in our EM algorithm can be computed using only signature counts (Section 2.2.4), so the input to McSplicer are the signature counts rather than individual reads.

We quantify the uncertainty of our estimator $\hat{\Theta}$ using bootstrapping. Specifically, let $c = (c_j)_{j=1}^J$ represent the signature counts over J signatures defined for a given gene, where the total signature count equals the total read count in the gene, i.e., $\sum_{j=1}^J c_j = N$. We draw B independent bootstrap samples, $c^1, \ldots, c^B$, from a multinomial distribution:

$$c^b \sim \text{multinomial}(\frac{c_1}{N}, \ldots, \frac{c_J}{N}, N). \tag{2.11}$$

Then, we compute B bootstrap estimators, $\hat{\Theta}^1, \ldots, \hat{\Theta}^B$, by applying our EM algorithm to each bootstrap sample and use them to approximate the sampling distribution of our estimator $\hat{\Theta}$. In this paper, we quantify the uncertainty of $\hat{\Theta}$ using a confidence interval computed from the approximated sampling distribution. Other types of uncertainty quantification could easily be obtained from the bootstrap estimators.

2.1.4 Simulated datasets and evaluation

We used Polyester (Frazee et al., 2015) to simulate reads from a human transcriptome with abundances estimated from a real RNA-experiment (GEO accession GSM3094221) using RSEM (Li and Dewey, 2011). Based on these ground truth expressions, we simulated data sets with varying sequencing depth commonly observed in practice, including 20 million, 50 million, and 75 million reads of 100bp length. Following the same strategy as Soneson et al. (2016), we randomly selected a set of 1000 genes with at least two expressed transcripts and sufficiently high ground truth

expression (gene-level read count per kilobase above 500). Among splice sites for which parameters estimated by compared methods have the same meaning (*comparable* splice sites, introduced in Section 2.3), we exclude from the analysis constitutive ones with true usage 1 and splice sites that are not used by any of the expressed transcripts (usage 0). That is, only splice sites that are alternatively used or not used by expressed transcripts are considered.

From the ground truth abundance of transcripts, we calculate the true usage of a splice site as the relative contribution of transcripts using a given splice site to the total expression of a gene. Specifically, let $A(s)$ and $B(s)$ denote subsets of transcripts in a gene G that either use or do not use a particular splice site s, respectively. Then the true usage of splice site s is computed by

$$u_s = \frac{\sum_{t \in A(s)} \theta_t}{\sum_{t \in A(s) \cup B(s)} \theta_t} = \frac{\sum_{t \in A(s)} \theta_t}{\sum_{t \in G} \theta_t}, \tag{2.12}$$

where θ_t represents the true abundance of transcript t.

We quantify the accuracy of splice site usages inferred by each method by using the Kullback-Leibler (KL) divergence. For a given splice site s, the two possible outcomes, whether or not a transcript uses the splice site can be modelled by a Bernoulli distribution with the splice site usage u_s, denoted by $Bernoulli(u_s)$. Let $\hat{u}_s$ represent the estimated splice usage. Then, we measure the accuracy of $\hat{u}_s$ using the KL divergence of $Bernoulli(\hat{u}_s)$ from $Bernoulli(u_s)$:

$$D_{KL}\left(Bernoulli(u_s) \| Bernoulli(\hat{u}_s)\right) = u_s \log \frac{u_s}{\hat{u}_s} + (1-u_s) \log \frac{1-u_s}{1-\hat{u}_s}. \tag{2.13}$$

All code and data necessary to reproduce the results of this simulation study are available at `https://github.com/canzarlab/McSplicer`.

2.2 Method details

After introducing necessary notation in Section 2.2.1, we will introduce the inhomogeneous Markov chain model of McSplicer in Section 2.2.2, present the likelihood of the model parameters in Section 2.2.3, describe the EM algorithm for estimating the parameters in Section 2.2.4, and provide a detailed description of algorithms to compute quantities used by the EM algorithm in Section 2.2.5.

2.2.1 Notations

In this section we introduce the notation used to describe our method McSplicer. As described in the main part of this work, we assume that exon start and end sites for a gene are given. This information can be obtained from known gene annotations or inferred from RNA-seq data using methods such as StringTie. Suppose we have M_s exon start sites $s_1, \ldots, s_{M_s}$, and M_e exon end sites $e_1, \ldots, e_{M_e}$, excluding the start site of the first exon and the end site of the last exon. All exon start and end sites partition the gene into M segments, $X_1, \ldots, X_M$, where $M = M_s + M_e + 1$. We introduce a sequence of hidden variables, $Z = (Z_1, \ldots, Z_M)$, where Z_i is an indicator for whether segment X_i is part of a transcript [1] ($Z_i = 1$) or not ($Z_i = 0$).

We define a subpath s by a sequence of states for $(Z_a, \ldots, Z_b)$, $1 \le a \le b \le M$. Specifically, a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$, where $o_i \in \{0,1\}$ for $i = a, \ldots, b$, is defined by $Z_a = o_a, Z_{a+1} =$

[1] We use terms transcript and isoform interchangeably to refer to a splice variant of a gene.

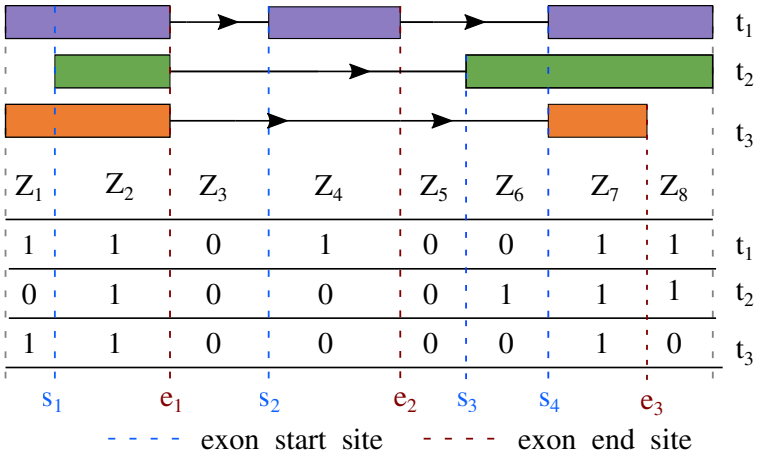


Figure 2.6: Hidden variables for segments defined by 3 different transcripts. The three sequences $(1,1,0,1,0,0,1,1),(0,1,0,0,0,1,1,1)$, and $(1,1,0,0,0,0,1,0)$ represent the three transcripts t_1,t_2, and t_3, respectively.

$o_{a+1},\ldots,Z_b = o_b$. In other words, a subpath $s = z_{[a:b]}(o_a,\ldots,o_b)$ describes whether each of the segments from X_a to X_b belongs to a transcript or not. Then, the probability of a subpath s is:

$$\Pr(z_{[a:b]}(o_a,\ldots,o_b)) = \Pr(Z_a = o_a, Z_{a+1} = o_{a+1},\ldots,Z_b = o_b), \tag{2.14}$$

which is given by our inhomogeneous Markov chain model. A path t is a subpath with $a = 1$ and $b = M$. A transcript can be represented by a path t, i.e., a sequence of states for $Z = (Z_1,\ldots,Z_M)$. Figure 2.6 shows an illustrative example of a gene with three transcripts which have four exon start sites and three exon end sites. These exon start and end sites divide the gene into eight segments ($M_s = 4$, $M_e = 3$, and $M = 8$). For example, a path $t = z_{[1:8]}(1,1,0,1,0,0,1,1)$ (i.e., $Z = (1,1,0,1,0,0,1,1)$) indicates transcript t_1, and a subpath $s = z_{[3:5]}(0,1,0)$ indicates a subpath obtained from the same transcript t_1.

We define the length of a subpath $s = z_{[a:b]}(o_a,\ldots,o_b)$, denoted by $l(s)$, by the number of bases included as part of a transcript:

$$l(s) = l(z_{[a:b]}(o_a,\ldots,o_b)) = \sum_{a\le i\le b: o_i=1} l(X_i), \tag{2.15}$$

where $l(X_j)$ is the number of bases in segment X_j. In the example of Figure 2.6, let us consider a subpath of the transcript t_1, $s = z_{[3:5]}(0,1,0)$. Then, $l(s) = l(z_{[3:5]}(0,1,0)) = l(X_4)$. Similarly, we can define the length of a transcript (or a path), denoted by $l(t)$, by the number of bases included in the exonic regions of that transcript:

$$l(t) = l(z_{[1:M]}(o_1,\ldots,o_M)) = \sum_{1\le i\le M: o_i=1} l(X_i). \tag{2.16}$$

In the example shown in Figure 2.6, transcript t_1 has length $l(t_1) = l(z_{[1:8]}(1,1,0,1,0,0,1,1)) = l(X_1)+l(X_2)+l(X_4)+l(X_7)+l(X_8)$.

We use $F(s)$ to denote the index of the first segment in a subpath s which is part of a transcript, and use $L(s)$ to denote the index of the last segment in a subpath s which is part of a transcript. In the example shown in Figure 2.6, let us consider a subpath of t_1, $s = z_{[3:5]}(0,1,0)$. Then, $F(s) = 4$

and $L(s) = 4$. For transcript t_1 path $t_1 = z_{[1:8]}(1,1,0,1,0,0,1,1)$, $F(t_1) = 1$ and $L(t_1) = 8$. For transcript t_2, path $t_2 = z_{[1:8]}(0,1,0,0,0,1,1,1)$, $F(t_2) = 2$ and $L(t_2) = 8$. Similarly, for transcript t_3 path $t_3 = z_{[1:8]}(1,1,0,0,0,0,1,0)$, $F(t_3) = 1$ and $L(t_3) = 7$.

2.2.2 An inhomogeneous Markov chain model

In this section, we describe an inhomogeneous Markov chain to model the relative abundance of transcripts. We assume that $Z = (Z_1, \ldots, Z_M)$ follows an inhomogeneous Markov chain. Specifically, for the first segment X_1,

$$\Pr(Z_1 = 1) = \pi. \tag{2.17}$$

For two consecutive segments X_i and X_{i+1} for $i = 1, \ldots, M-1$ that are separated by exon start site s_m for $m = 1, \ldots, M_s$ (i.e., $i = I(s_m)$, where $I(s_m)$ is the index of the segment which appears on the left side of exon start site s_m),

$$\Pr(Z_{i+1} = 1 | Z_i = 0) = p_m, \tag{2.18}$$
$$\Pr(Z_{i+1} = 1 | Z_i = 1) = 1. \tag{2.19}$$

If they are separated by exon end site e_m for $m = 1, \ldots, M_e$ (i.e., $i = I(e_m)$, where $I(e_m)$ is the index of the segment which appears on the left side of exon end site e_m),

$$\Pr(Z_{i+1} = 0 | Z_i = 0) = 1, \tag{2.20}$$
$$\Pr(Z_{i+1} = 0 | Z_i = 1) = q_m. \tag{2.21}$$

With this transition probability, we do not allow transcripts where $Z_i = 1$ and $Z_{i+1} = 0$ for $i = I(s_m)$, or $Z_i = 0$ and $Z_{i+1} = 1$ for $i = I(e_m)$. The parameters $p = (p_1, \ldots, p_{M_s})$ and $q = (q_1, \ldots, q_{M_e})$ indicate probabilities of using exon start sites and exon end sites, respectively. Precisely, these are conditional probabilities given that each site is considered for potential use. For example, with the current segment being part of a transcript (i.e., $Z_i = 1$), the splicing process ignores an exon start site (i.e., $\Pr(Z_{i+1} = 1 | Z_i = 1) = 1$ if $i = I(s_m)$) while it considers an exon end site for potential use (i.e., $\Pr(Z_{i+1} = 0 | Z_i = 1) = q_m$ if $i = I(e_m)$). Table 2.1 lists probabilities for the three transcripts (or paths) in Figure 2.6 under our Markov model. Furthermore, to handle different transcript start and end sites within a gene, we introduce artificial starting and end points (i.e., reference points) in the implementation of this model.

2.2.3 Likelihood of the parameters $\Theta = (\pi, p_1, \ldots, p_{M_s}, q_1, \ldots, q_{M_e})$

In this section we present the likelihood of the model parameters. Suppose we have RNA-seq reads mapped to a particular gene. The reads are derived from one end of each of the N fragments and each read has length L. We assume that each fragment is independently generated from one of the possible transcripts allowed by our model. We denote the sequence of the n-th read as r_n. T_n represents the transcript from which r_n was generated. S_n denotes the shortest subpath of T_n from which r_n is derived. B_n denotes the start position of r_n in T_n. For example, Figure 2.7 shows that r_n was derived from the first transcript (i.e., $T_n = t_1$), thus $T_n = z_{[1:8]}(1,1,0,1,0,0,1,1)$. The shortest subpath of T_n from which read n was derived is $S_n = z_{[2:4]}(1,0,1)$.

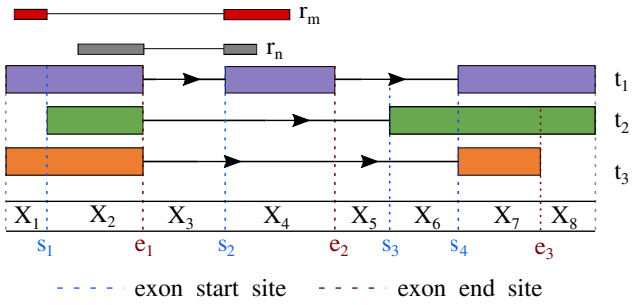


Figure 2.7: An example of a gene with three transcripts, the same as the one shown in Figure 2.6. Here, read r_n was derived from the first transcript ($T_n = t_1$) and is compatible with our model. In contrast, r_m is not compatible with our model since the two segments X_1 and X_2 are separated by exon start site s_1 and thus our model does not allow $S_m = z_{[1:4]}(1,0,0,1)$.

Assuming all r_n are derived from transcripts that are allowed in our model (i.e., $\Pr(r_n) > 0$ for all r_n), we remove reads that are not compatible with our model (see Figure 2.7). The likelihood of Θ can be written as:

$$\begin{aligned}
\Pr(r|\Theta) &= \prod_{n=1}^{N} \Pr(r_n|\Theta) \\
&= \prod_{n=1}^{N} \Big[\sum_t \Pr(r_n, T_n = t|\Theta)\Big] \\
&= \prod_{n=1}^{N} \Big[\sum_t \Big[\sum_{(s,b):s\subset t} \Pr(r_n, S_n = s, B_n = b, T_n = t|\Theta)\Big]\Big] \\
&\quad \text{where } s \subset t \text{ means } s \text{ is a subpath of } t, \\
&= \prod_{n=1}^{N} \Big[\sum_t \Big[\sum_{(s,b):s\subset t} \Pr(r_n|S_n = s, B_n = b) \Pr(S_n = s, B_n = b|T_n = t) \Pr(T_n = t|\Theta)\Big]\Big] \\
&= \prod_{n=1}^{N} \Big[\sum_t \Big[\sum_{(s,b):s\subset t,(s,b)\to r_n} 1 \frac{1}{l(t)} \frac{l(t) w_\Theta(t)}{D(\Theta)}\Big]\Big] \\
&\quad \text{where } (s,b) \to r_n \text{ denotes that } r_n \text{ is the length } L \text{ sequence starting at position } b \\
&\quad \text{in the concatenation of segments in } s, \\
&= \prod_{n=1}^{N} \Big[\sum_t \Big[\sum_{(s,b):s\subset t,(s,b)\to r_n} \frac{w_\Theta(t)}{D(\Theta)}\Big]\Big],
\end{aligned} \tag{2.22}$$

where $D(\Theta) = \sum_t l(t) w_\Theta(t)$. $l(t)$ represents the (effective) length (Trapnell et al., 2010) of transcript t, and $w_\Theta(t)$ represents the relative frequency (probability) of transcript t.

2.2.4 Parameter estimation using the EM algorithm

We use an EM algorithm to compute the maximum likelihood estimate for the model parameters $\Theta = \{\pi, p, q\}$, that is, $\hat{\Theta} := \text{argmax}_\Theta \Pr(r|\Theta)$. In this section we describe the EM-steps to obtain the MLE for our model parameters. Let $Z^n = (Z_1^n, \ldots, Z_M^n)$ represent the isoform T_n, that

is, the path from which read n was derived. Then, the complete data likelihood, $\Pr(r,Z|\Theta) = \prod_{n=1}^{N}\Pr(r_n,Z^n|\Theta)$, can be written as

$$\begin{aligned}
&\prod_{n=1}^{N}\Big[\sum_{(s,b):s\subset Z^n}\Pr(r_n,s_n=s,b_n=b,Z^n|\Theta)\Big]\\
&\text{where } s\subset Z^n \text{ means } s \text{ is a subpath of the path } Z^n,\\
&=\prod_{n=1}^{N}\Big[\sum_{(s,b):s\subset Z^n}\Pr(r_n|s_n=s,b_n=b)\Pr(s_n=s,b_n=b|Z^n)\Pr(Z^n|\Theta)\Big]\\
&=\prod_{n=1}^{N}\Big[\sum_{(s,b):s\subset Z^n,(s,b)\to r_n}1\frac{1}{l(Z^n)}\frac{l(Z^n)w_\Theta(Z^n)}{D(\Theta)}\Big]\\
&=\prod_{n=1}^{N}\Big[\sum_{(s,b):s\subset Z^n,(s,b)\to r_n}\frac{w_\Theta(Z^n)}{D(\Theta)}\Big]\\
&=\prod_{n=1}^{N}\Big[\frac{C(r_n,Z^n)w_\Theta(Z^n)}{D(\Theta)}\Big]
\end{aligned}\tag{2.23}$$

where $C(r_n,Z^n)$ indicates the number of (s,b) in the isoform T_n (defined by Z^n) which are matched to r_n. Then, we can rewrite it as

$$\begin{aligned}
&\frac{1}{D(\Theta)^N}\prod_{n=1}^{N}\Big[C(r_n,Z^n)\big[\pi^{Z_1^n}(1-\pi)^{1-Z_1^n}\big]\\
&\times\Big[\prod_{m=1}^{M_s}p_m^{(1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1})}(1-p_m)^{(1-Z^n_{I(s_m)})(1-Z^n_{I(s_m)+1})}\Big]\\
&\times\Big[\prod_{m=1}^{M_e}q_m^{(Z^n_{I(e_m)})(1-Z^n_{I(e_m)+1})}(1-q_m)^{(Z^n_{I(e_m)})(Z^n_{I(e_m)+1})}\Big]\Big].
\end{aligned}\tag{2.24}$$

And we can write a log likelihood $\log\Pr(r,Z|\Theta)$ as

$$\begin{aligned}
&-N\log D(\Theta)+\sum_{n=1}^{N}\log C(r_n,Z^n)+\sum_{n=1}^{N}Z_1^n\log\pi+\sum_{n=1}^{N}(1-Z_1^n)\log(1-\pi)\\
&+\sum_{n=1}^{N}\sum_{m=1}^{M_s}\Big[(1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1})\log p_m\Big]+\sum_{n=1}^{N}\sum_{m=1}^{M_s}\Big[(1-Z^n_{I(s_m)})(1-Z^n_{I(s_m)+1})\log(1-p_m)\Big]\\
&+\sum_{n=1}^{N}\sum_{m=1}^{M_e}\Big[(Z^n_{I(e_m)})(1-Z^n_{I(e_m)+1})\log q_m\Big]+\sum_{n=1}^{N}\sum_{m=1}^{M_e}\Big[(Z^n_{I(e_m)})(Z^n_{I(e_m)+1})\log(1-q_m)\Big].
\end{aligned}\tag{2.25}$$

Note that the transition probabilities in our model do not allow isoforms where $Z_{I(s_m)}=1$ and $Z_{I(s_m)+1}=0$ at any exon start site and $Z_{I(e_m)}=0$ and $Z_{I(e_m)+1}=1$ at any exon end site, and $C(r_n,Z^n)$ does not depend on Θ.

M-step

Let $\Theta^l = (\pi^l, p_1^l, \ldots, p_{M_s}^l, q_1^l, \ldots, q_{M_e}^l)$ represent the model parameter values at the l-th iteration of the EM algorithm. Then, new parameter estimates at the $(l+1)$-th iteration are the values of Θ which maximize $Q(\Theta|\Theta^l) := E_{Z|r,\Theta^l}[\log \Pr(r, Z|\Theta)]$. Let $\Theta^{l+1} = (\pi^{l+1}, p_1^{l+1}, \ldots, p_{M_s}^{l+1}, q_1^{l+1}, \ldots, q_{M_e}^{l+1})$ denote the parameter estimates at the $(l+1)$-th iteration, then

$$\begin{aligned}\Theta^{l+1} &= argmax_\Theta Q(\Theta|\Theta^l), \\ &= argmax_\Theta E_{Z|r,\Theta^l}[\log \Pr(r, Z|\Theta)].\end{aligned} \tag{2.26}$$

We will describe how to compute Θ^{l+1} in Section 2.2.4 (for $p_1^{l+1}, \ldots, p_{M_s}^{l+1}$), Section 2.2.4 (for $q_1^{l+1}, \ldots, q_{M_e}^{l+1}$), and Section 2.2.4 (for π^{l+1}).

p_m^{l+1} **for** $m = 1, \ldots, M_s$

Let $p'_m = 1 - p_m$ for $m = 1, \ldots, M_s$, $q'_m = 1 - q_m$ for $m = 1, \ldots, M_e$, and $\pi' = 1 - \pi$. Then, the Lagrangian function for maximizing $Q(\Theta|\Theta^l)$ is proportional to

$$\begin{aligned}\Lambda = &-N \log D(\Theta) + \sum_{n=1}^{N} \Pr(Z_1^n = 1|r_n, \Theta^l) \log \pi + \sum_{n=1}^{N} \Pr(Z_1^n = 0|r_n, \Theta^l) \log \pi' \\ &+ \sum_{n=1}^{N} \sum_{m=1}^{M_s} \left[\Pr((1 - Z_{I(s_m)}^n)(Z_{I(s_m)+1}^n) = 1|r_n, \Theta^l) \log p_m \right] \\ &+ sum_{n=1}^{N} \sum_{m=1}^{M_s} \left[\Pr((1 - Z_{I(s_m)}^n)(1 - Z_{I(s_m)+1}^n) = 1|r_n, \Theta^l) \log p'_m \right] \\ &+ \sum_{n=1}^{N} \sum_{m=1}^{M_e} \left[\Pr((Z_{I(e_m)}^n)(1 - Z_{I(e_m)+1}^n) = 1|r_n, \Theta^l) \log q_m \right] \\ &+ \sum_{n=1}^{N} \sum_{m=1}^{M_e} \left[\Pr((Z_{I(e_m)}^n)(Z_{I(e_m)+1}^n) = 1|r_n, \Theta^l) \log q'_m \right] \\ &- \lambda^\pi(\pi + \pi' - 1) - \sum_{m=1}^{M_s} \lambda_m^s (p_m + p'_m - 1) - \sum_{m=1}^{M_e} \lambda_m^e (q_m + q'_m - 1).\end{aligned} \tag{2.27}$$

We take derivatives with respect to p_m and p'_m for $m = 1, \ldots, M_s$ and set them to zero, leading to

$$\begin{aligned}\frac{\partial}{\partial p_m}\Lambda &= -N \frac{1}{D(\Theta)} \frac{\partial D(\Theta)}{\partial p_m} + \frac{1}{p_m} \sum_{n=1}^{N} \Pr((1 - Z_{I(s_m)}^n)(Z_{I(s_m)+1}^n) = 1|r_n, \Theta^l) - \lambda_m^s = 0, \\ \frac{\partial}{\partial p'_m}\Lambda &= -N \frac{1}{D(\Theta)} \frac{\partial D(\Theta)}{\partial p'_m} + \frac{1}{p'_m} \sum_{n=1}^{N} \Pr((1 - Z_{I(s_m)}^n)(1 - Z_{I(s_m)+1}^n) = 1|r_n, \Theta^l) - \lambda_m^s = 0.\end{aligned} \tag{2.28}$$

As $D(\Theta) = \sum_t l(t) w_\Theta(t) = E(l(T)) = E(l(Z))$ depends on Θ, it is difficult to find solutions for these equations. Borrowing an idea from LeGault and Dewey (2013), we use the fixed point iteration to solve for Θ. Thus, $\lambda_m^s = 0$ for $m = 1, \ldots, M_s$ and the fixed point iteration uses the equation

$$p_m^{l+1} = p_m = \frac{A_m}{A_m + B_m}, \tag{2.29}$$

where

$$A_m = \frac{\sum_{n=1}^{N} \Pr((1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1}) = 1 | r_n, \Theta^l)}{E(l(Z_{[1:I(s_m)]})|Z_{I(s_m)} = 0) + E(l(Z_{[I(s_m)+1:M]})|Z_{I(s_m)+1} = 1)},$$
$$B_m = \frac{\sum_{n=1}^{N} \Pr((1-Z^n_{I(s_m)})(1-Z^n_{I(s_m)+1}) = 1 | r_n, \Theta^l)}{E(l(Z_{[1:I(s_m)]})|Z_{I(s_m)} = 0) + E(l(Z_{[I(s_m)+1:M]})|Z_{I(s_m)+1} = 0)}, \tag{2.30}$$

and $Z_{[i:j]}$ for $i \leq j$ denote a subpath $(Z_i, \ldots, Z_j)$.

Remark 1: p_m^{l+1} can be computed using only signature counts instead of individual reads. Let $c = (c_j)_{j=1}^J$ represent the signature counts over J signatures. Reads mapping to the same signature have the same subpath for S_n (i.e., the shortest subpath of T_n from which read n is derived). Suppose r_n and $r_{n'}$ are reads mapping to the same j-th signature and s_j represents a subpath corresponding to the j-th signature. Then,

$$\begin{aligned}
\Pr((1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1}) = 1 | r_n, \Theta^l) &= \Pr((1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1}) = 1 | S_n = s_j, \Theta^l) \\
&= \Pr((1-Z^{n'}_{I(s_m)})(Z^{n'}_{I(s_m)+1}) = 1 | S_{n'} = s_j, \Theta^l) \\
&= \Pr((1-Z^{n'}_{I(s_m)})(Z^{n'}_{I(s_m)+1}) = 1 | r_{n'}, \Theta^l).
\end{aligned} \tag{2.31}$$

Instead of computing $\Pr((1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1}) = 1 | r_n, \Theta^l)$ for all reads r_n, we can compute them using s_j for $j = 1, \ldots, J$. Therefore, A_m (and analogously B_m) can be computed using only signature counts.

Remark 2: In the E-step (Section 2.2.4) we compute $\Pr((1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1}) = 1 | S_n = s_j, \Theta^l)$ and $\Pr((1-Z^n_{I(s_m)})(1-Z^n_{I(s_m)+1}) = 1 | S_n = s_j, \Theta^l)$ for $j = 1, \ldots, J$.

Remark 3: Sections 2.2.5 and 2.2.5 provide more detailed explanations of quantities $E(l(Z_{[1:I(s_m)]})|Z_{I(s_m)} = 0)$, $E(l(Z_{[I(s_m)+1:M]})|Z_{I(s_m)+1} = 1)$, $E(l(Z_{[1:I(s_m)]})|Z_{I(s_m)} = 0)$, and $E(l(Z_{[I(s_m)+1:M]})|Z_{I(s_m)+1} = 0)$, and describe how to compute them using dynamic programming.

q_m^{l+1} **for** $m = 1, \ldots, M_e$

Using a derivation similar to one for p_m^{l+1} above, we can obtain the following result. Let

$$C_m = \frac{\sum_{n=1}^{N} \Pr((Z^n_{I(e_m)})(1-Z^n_{I(e_m)+1}) = 1 | r_n, \Theta^l)}{E(l(Z_{[1:I(e_m)]})|Z_{I(e_m)} = 1) + E(l(Z_{[I(e_m)+1:M]})|Z_{I(e_m)+1} = 0)}, \tag{2.32}$$
$$D_m = \frac{\sum_{n=1}^{N} \Pr((Z^n_{I(e_m)})(Z^n_{I(e_m)+1}) = 1 | r_n, \Theta^l)}{E(l(Z_{[1:I(e_m)]})|Z_{I(e_m)} = 1) + E(l(Z_{[I(e_m)+1:M]})|Z_{I(e_m)+1} = 1)}. \tag{2.33}$$

Then

$$q_m^{l+1} = \frac{C_m}{C_m + D_m}. \tag{2.34}$$

Remark 1: Using a derivation similar to the one for p_m^{l+1} above, we can show that q_m^{l+1} can be computed using only signature counts.

Remark 2: In the E-step (Section 2.2.4) we compute $\Pr((Z^n_{I(e_m)})(1-Z^n_{I(e_m)+1})=1|S_n=s_j,\Theta^l)$ and $\Pr((Z^n_{I(e_m)})(Z^n_{I(e_m)+1})=1|S_n=s_j,\Theta^l)$.

Remark 3: Sections 2.2.5 and 2.2.5 provide more detailed explanations of quantities $E(l(Z_{[1:I(e_m)]})|Z_{I(e_m)}=1)$, $E(l(Z_{[I(e_m)+1:M]})|Z_{I(e_m)+1}=0)$, $E(l(Z_{[1:I(e_m)]})|Z_{I(e_m)}=1)$, and $E(l(Z_{[I(e_m)+1:M]})|Z_{I(e_m)+1}=1)$, and describe how to compute them using dynamic programming.

π^{l+1}

Using a derivation similar to one for p_m^{l+1} above, we can obtain the following result. Let

$$E=\frac{\sum_{n=1}^N \Pr(Z_1^n=1|r_n,\Theta^l)}{E(l(Z_{[1:M]})|Z_1=1)}, \tag{2.35}$$

$$F=\frac{\sum_{n=1}^N \Pr(Z_1^n=0|r_n,\Theta^l)}{E(l(Z_{[1:M]})|Z_1=0)}. \tag{2.36}$$

Then

$$\pi^{l+1}=\frac{E}{E+F}. \tag{2.37}$$

Remark 1: Using a derivation similar to the one for p_m^{l+1} above, we can show that π^{l+1} can be computed using only signature counts.

Remark 2: In the E-step (Section 2.2.4) we compute $\Pr(Z_1^n=1|S_n=s_j,\Theta^l)$ and $\Pr(Z_1^n=0|S_n=s_j,\Theta^l)$.

Remark 3: Section 2.2.5 provides more detailed explanations of quantities $E(l(Z_{[1:M]})|Z_1=1)$ and $E(l(Z_{[1:M]})|Z_1=0)$, and describes how to compute them using dynamic programming.

E-step

Let $c=(c_j)_{j=1}^J$ represent the signature counts over J signatures and s_j represents a subpath corresponding to the j-th signature.

$$\begin{aligned}\Pr((1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1})=1|S_n=s_j,\Theta^l)&=\frac{\Pr((1-Z^n_{I(s_m)})(Z^n_{I(s_m)+1})=1,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\\&=\frac{\Pr(Z^n_{I(s_m)}=0,Z^n_{I(s_m)+1}=1,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\end{aligned} \tag{2.38}$$

$$\begin{aligned}\Pr((1-Z^n_{I(s_m)})(1-Z^n_{I(s_m)+1})=1|S_n=s_j,\Theta^l) &= \frac{\Pr((1-Z^n_{I(s_m)})(1-Z^n_{I(s_m)+1})=1,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\\ &= \frac{\Pr(Z^n_{I(s_m)}=0,Z^n_{I(s_m)+1}=0,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\end{aligned} \tag{2.39}$$

$$\begin{aligned}\Pr((Z^n_{I(e_m)})(Z^n_{I(e_m)+1})=1|S_n=s_j,\Theta^l) &= \frac{\Pr((Z^n_{I(e_m)})(Z^n_{I(e_m)+1})=1,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\\ &= \frac{\Pr(Z^n_{I(e_m)}=1,Z^n_{I(e_m)+1}=1,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\end{aligned} \tag{2.40}$$

$$\begin{aligned}\Pr((Z^n_{I(e_m)})(1-Z^n_{I(e_m)+1})=1|S_n=s_j,\Theta^l) &= \frac{\Pr((Z^n_{I(e_m)})(1-Z^n_{I(e_m)+1})=1,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\\ &= \frac{\Pr(Z^n_{I(e_m)}=1,Z^n_{I(e_m)+1}=0,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)},\end{aligned} \tag{2.41}$$

$$\Pr(Z^n_1=1|S_n=s_j,\Theta^l)=\frac{\Pr(Z^n_1=1,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)}, \tag{2.42}$$

$$\Pr(Z^n_1=0|S_n=s_j,\Theta^l)=\frac{\Pr(Z^n_1=0,S_n=s_j|\Theta^l)}{\Pr(S_n=s_j|\Theta^l)}, \tag{2.43}$$

where

$$\begin{aligned}&\Pr(S_n=s_j|\Theta^l)=\Pr(Z^n_1=0,S_n=s_j|\Theta^l)+\Pr(Z^n_1=1,S_n=s_j|\Theta^l)\\ &=\Pr(Z^n_1=0)\Pr(Z^n_{F(s_j)}=1|Z^n_1=0)\Pr(S_n=s_j|Z^n_{F(s_j)}=1)\\ &+\Pr(Z^n_1=1)\Pr(Z^n_{F(s_j)}=1|Z^n_1=1)\Pr(S_n=s_j|Z^n_{F(s_j)}=1)\\ &=(1-\pi)\Pr(Z^n_{F(s_j)}=1|Z^n_1=0)\Pr(S_n=s_j|Z^n_{F(s_j)}=1)+\pi\Pr(Z^n_{F(s_j)}=1|Z^n_1=1)\Pr(S_n=s_j|Z^n_{F(s_j)}=1).\end{aligned} \tag{2.44}$$

Remark 1: We describe how to compute $\Pr(S_n=s_j|Z^n_{F(s_j)}=1)$ in Section 2.2.5, $\Pr(Z^n_{I(s_m)}=0,Z^n_{I(s_m)+1}=1,S_n=s_j|\Theta^l)$ in Section 2.2.5, $\Pr(Z^n_{I(s_m)}=0,Z^n_{I(s_m)+1}=0,S_n=s_j|\Theta^l)$ in Section 2.2.5, $\Pr(Z^n_{I(e_m)}=1,Z^n_{I(e_m)+1}=1,S_n=s_j|\Theta^l)$ in Section 2.2.5, $\Pr(Z^n_{I(e_m)}=1,Z^n_{I(e_m)+1}=0,S_n=s_j|\Theta^l)$ in Section 2.2.5, and $\Pr(Z^n_1=1,S_n=s_j|\Theta^l)$ and $\Pr(Z^n_1=0,S_n=s_j|\Theta^l)$ in Section 2.2.5.

Remark 2: In Section 2.2.5 we describe the dynamic programming algorithm to compute $\Pr(Z^n_{F(s_j)}=1|Z^n_1=0)$ and $\Pr(Z^n_{F(s_j)}=1|Z^n_1=1)$.

Remark 3: $\Pr(Z^n_1=0|S_n=s_j,\Theta^l)$ can also be computed as $1-\Pr(Z^n_1=1|S_n=s_j,\Theta^l)$.

2.2.5 Computation of quantities used by the EM algorithm

In this section we provide a detailed description of algorithms to compute quantities used by the EM algorithm introduced in Section 2.2.4. Some quantities can be computed efficiently using dynamic programming (DP).

Expected prefix lengths: $l_p(i,\mathbf{in}) := E(l(Z_{[1:i]})|Z_i = 1)$ and $l_p(i,\mathbf{out}) := E(l(Z_{[1:i]})|Z_i = 0)$ for the i-th segment

The expected prefix lengths have been used in the M-step of the EM algorithm (see Section 2.2.4). In this section we formally define them and describe how to compute them using dynamic programming.

Definition

We define two types of the expected prefix length for the i-th segment, $l_p(i,\mathrm{in})$ and $l_p(i,\mathrm{out})$, as follows. Let $Z_{[1:i]}$ denote a subpath which describes a sequence of states for $(Z_1,\ldots,Z_i)$. Then, the length of the subpath $Z_{[1:i]}$ is given by

$$l(Z_{[1:i]}) = \sum_{1\leq j\leq i: Z_j=1} l(X_j), \tag{2.45}$$

where $l(X_j)$ indicates the number of exonic bases in the segment X_j. $l_p(i,\mathrm{in})$ is defined by the expected length of the subpath $Z_{[1:i]}$ given that X_i is a part of a transcript (i.e., $Z_i = 1$) and $l_p(i,\mathrm{out})$ is defined by the expected length of the subpath $Z_{[1:i]}$ given that X_i is not a part of a transcript (i.e., $Z_i = 0$). Specifically,

$$l_p(i,\mathrm{in}) = E(l(Z_{[1:i]})|Z_i = 1) \tag{2.46}$$
$$l_p(i,\mathrm{out}) = E(l(Z_{[1:i]})|Z_i = 0). \tag{2.47}$$

Computing expected prefix lengths by dynamic programming

We can compute the expected prefix lengths using dynamic programming as follows.

For $i = 1$,

$$l_p(1,\mathrm{in}) = l(X_1) \tag{2.48}$$
$$l_p(1,\mathrm{out}) = 0. \tag{2.49}$$

For $i = 2, \ldots, M$,

$$\begin{aligned}
l_p(i,\text{in}) &= E(l(Z_{[1:i]})|Z_i = 1) \\
&= l(X_i) + E(l(Z_{[1:(i-1)]}), Z_{i-1} = 1|Z_i = 1) + E(l(Z_{[1:(i-1)]}), Z_{i-1} = 0|Z_i = 1) \\
&= l(X_i) + E(l(Z_{[1:(i-1)]})|Z_{i-1} = 1, Z_i = 1)\Pr(Z_{i-1} = 1|Z_i = 1) \\
&+ E(l(Z_{[1:(i-1)]})|Z_{i-1} = 0, Z_i = 1)\Pr(Z_{i-1} = 0|Z_i = 1) \\
&\text{because } Z_{[1:(i-1)]} \text{ and } Z_i \text{ are independent conditional on } Z_{i-1} \\
&= l(X_i) + E(l(Z_{[1:(i-1)]})|Z_{i-1} = 1)\Pr(Z_{i-1} = 1|Z_i = 1) \\
&+ E(l(Z_{[1:(i-1)]})|Z_{i-1} = 0)\Pr(Z_{i-1} = 0|Z_i = 1) \\
&= l(X_i) + l_p(i-1,\text{in})\frac{\Pr(Z_{i-1} = 1)\Pr(Z_i = 1|Z_{i-1} = 1)}{\Pr(Z_i = 1)} \\
&+ l_p(i-1,\text{out})\frac{\Pr(Z_{i-1} = 0)\Pr(Z_i = 1|Z_{i-1} = 0)}{\Pr(Z_i = 1)}.
\end{aligned} \tag{2.50}$$

Similarly,

$$\begin{aligned}
l_p(i,\text{out}) &= E(l(Z_{[1:i]})|Z_i = 0) \\
&= E(l(Z_{[1:(i-1)]}), Z_{i-1} = 1|Z_i = 0) + E(l(Z_{[1:(i-1)]}), Z_{i-1} = 0|Z_i = 0) \\
&= l_p(i-1,\text{in})\frac{\Pr(Z_{i-1} = 1)\Pr(Z_i = 0|Z_{i-1} = 1)}{\Pr(Z_i = 0)} \\
&+ l_p(i-1,\text{out})\frac{\Pr(Z_{i-1} = 0)\Pr(Z_i = 0|Z_{i-1} = 0)}{\Pr(Z_i = 0)}.
\end{aligned} \tag{2.51}$$

Remark 1: If segments X_{i-1} and X_i are separated by exon start site s_m (i.e., $i-1 = I(s_m)$),

$$\Pr(Z_i = 1|Z_{i-1} = 0) = p_m \tag{2.52}$$
$$\Pr(Z_i = 1|Z_{i-1} = 1) = 1, \tag{2.53}$$
$$\Pr(Z_i = 0|Z_{i-1} = 0) = 1 - p_m \tag{2.54}$$
$$\Pr(Z_i = 0|Z_{i-1} = 1) = 0, \tag{2.55}$$

and if segments X_{i-1} and X_i are separated by exon end site e_m (i.e., $i-1 = I(e_m)$),

$$\Pr(Z_i = 1|Z_{i-1} = 0) = 0 \tag{2.56}$$
$$\Pr(Z_i = 1|Z_{i-1} = 1) = 1 - q_m, \tag{2.57}$$
$$\Pr(Z_i = 0|Z_{i-1} = 0) = 1 \tag{2.58}$$
$$\Pr(Z_i = 0|Z_{i-1} = 1) = q_m. \tag{2.59}$$

Remark 2: Section 2.2.5 describes the dynamic programming algorithm to compute $\Pr(Z_i = 0)$ and $\Pr(Z_i = 1)$ for $i = 1, \ldots, M$.

Expected suffix lengths: $l_s(i,\mathbf{in}) := E(l(Z_{[i:M]})|Z_i = 1)$ and $l_s(i,\mathbf{out}) := E(l(Z_{[i:M]})|Z_i = 0)$ for the $i-th$ segment

The expected suffix lengths have been used in the M-step of the EM algorithm (see Section 2.2.4). In this section we formally define them and describe how to compute them using dynamic programming.

Definition

We define two types of expected suffix length for the $i-th$ segment, $l_s(i,\text{in})$ and $l_s(i,\text{out})$, as follows. Let $Z_{[i:M]}$ denote a subpath which describes a sequence of states for $(Z_i,\ldots,Z_M)$. $l_s(i,\text{in})$ is defined by the expected length of the subpath $Z_{[i:M]}$ given that X_i is a part of an isoform (i.e., $Z_i = 1$) and $l_s(i,\text{out})$ is defined by the expected length of the subpath $Z_{[i:M]}$ given that X_i is not a part of an isoform (i.e., $Z_i = 0$). Specifically,

$$l_s(i,\text{in}) = E(l(Z_{[i:M]})|Z_i = 1) \tag{2.60}$$

$$l_s(i,\text{out}) = E(l(Z_{[i:M]})|Z_i = 0). \tag{2.61}$$

Computing expected suffix lengths by dynamic programming

We can compute the expected suffix lengths using dynamic programming as follows.

For $i = 1,\ldots,M-1$,

$$\begin{aligned}
l_s(i,\text{in}) &= E(l(Z_{[i:M]})|Z_i = 1) \\
&= l(X_i) + E(l(Z_{[(i+1):M]}), Z_{i+1} = 1|Z_i = 1) + E(l(Z_{[(i+1):M]}), Z_{i+1} = 0|Z_i = 1) \\
&= l(X_i) + E(l(Z_{[(i+1):M]})|Z_{i+1} = 1, Z_i = 1)\Pr(Z_{i+1} = 1|Z_i = 1) \\
&+ E(l(Z_{[(i+1):M]})|Z_{i+1} = 0, Z_i = 1)\Pr(Z_{i+1} = 0|Z_i = 1) \\
&\text{because } Z_{[(i+1):M]} \text{ and } Z_i \text{ are independent conditional on } Z_{i+1} \\
&= l(X_i) + E(l(Z_{[(i+1):M]})|Z_{i+1} = 1)\Pr(Z_{i+1} = 1|Z_i = 1) \\
&+ E(l(Z_{[(i+1):M]})|Z_{i+1} = 0)\Pr(Z_{i+1} = 0|Z_i = 1) \\
&= l(X_i) + l_s(i+1,\text{in})\Pr(Z_{i+1} = 1|Z_i = 1) \\
&+ l_s(i+1,\text{out})\Pr(Z_{i+1} = 0|Z_i = 1).
\end{aligned} \tag{2.62}$$

Similarly

$$\begin{aligned}
l_s(i,\text{out}) &= E(l(Z_{[i:M]})|Z_i = 0) \\
&= E(l(Z_{[(i+1):M]}), Z_{i+1} = 1|Z_i = 0) + E(l(Z_{[(i+1):M]}), Z_{i+1} = 0|Z_i = 0) \\
&= E(l(Z_{[(i+1):M]})|Z_{i+1} = 1, Z_i = 0)\Pr(Z_{i+1} = 1|Z_i = 0) \\
&+ E(l(Z_{[(i+1):M]})|Z_{i+1} = 0, Z_i = 0)\Pr(Z_{i+1} = 0|Z_i = 0) \\
&= l_s(i+1,\text{in})\Pr(Z_{i+1} = 1|Z_i = 0) \\
&+ l_s(i+1,\text{out})\Pr(Z_{i+1} = 0|Z_i = 0).
\end{aligned} \tag{2.63}$$

And for $i = M$,

$$l_s(M,\text{in}) = l(X_M) \tag{2.64}$$

$$l_s(M, \text{out}) = 0. \tag{2.65}$$

Remark 1: For the computation of $\Pr(Z_i = 1|Z_{i-1} = 0)$, $\Pr(Z_i = 1|Z_{i-1} = 1)$, $\Pr(Z_i = 0|Z_{i-1} = 0)$, and $\Pr(Z_i = 0|Z_{i-1} = 1)$, see Remark 1 in Section 2.2.5.

Computing $\Pr(Z_i = 1)$ and $\Pr(Z_i = 0)$ using dynamic programming

The probability that segment X_i is part of a transcript, $\Pr(Z_i = 1)$, and the probability that segment X_i is not part of a transcript, $\Pr(Z_i = 0)$, have been used in the dynamic program to compute the expected prefix lengths in Section 2.2.5. We can compute $\Pr(Z_i = 1)$ and $\Pr(Z_i = 0)$ using dynamic programming as follows.

For $i = 1$,

$$\Pr(Z_1 = 0) = 1 - \pi, \tag{2.66}$$
$$\Pr(Z_1 = 1) = \pi. \tag{2.67}$$

For $i = 2, \ldots, M$,

$$\Pr(Z_i = 1) = \Pr(Z_{i-1} = 1)\Pr(Z_i = 1|Z_{i-1} = 1) + \Pr(Z_{i-1} = 0)\Pr(Z_i = 1|Z_{i-1} = 0), \tag{2.68}$$

$$\begin{aligned}\Pr(Z_i = 0) &= 1 - \Pr(Z_i = 1), \text{ or equivalently} \\ &= \Pr(Z_{i-1} = 1)\Pr(Z_i = 0|Z_{i-1} = 1) + \Pr(Z_{i-1} = 0)\Pr(Z_i = 0|Z_{i-1} = 0),\end{aligned} \tag{2.69}$$

Remark 1: For the computation of $\Pr(Z_i = 1|Z_{i-1} = 0)$, $\Pr(Z_i = 1|Z_{i-1} = 1)$, $\Pr(Z_i = 0|Z_{i-1} = 0)$, and $\Pr(Z_i = 0|Z_{i-1} = 1)$, see Remark 1 in Section 2.2.5.

Computing $\Pr(Z_j = 1|Z_i = 1)$, $\Pr(Z_j = 0|Z_i = 1)$, $\Pr(Z_j = 1|Z_i = 0)$, and $\Pr(Z_j = 0|Z_i = 0)$ for $1 \le i \le j \le M$ using dynamic programming

The E-step in Section 2.2.4 used $\Pr(Z_j^n = 1|Z_1^n = 0)$ and $\Pr(Z_j^n = 1|Z_1^n = 0)$ for $j = 1, \ldots, M$ to compute $\Pr(S_n = s_j|\Theta^l)$. We also use $\Pr(Z_j^n = 1|Z_i^n = 1)$, $\Pr(Z_j^n = 0|Z_i^n = 1)$, $\Pr(Z_j^n = 1|Z_i^n = 0)$, and $\Pr(Z_j^n = 0|Z_i^n = 0)$ for $1 \le i \le j \le M$ in Sections 2.2.5, 2.2.5, 2.2.5, 2.2.5, and 2.2.5. Here, we describe their computation using dynamic programming (DP). As these quantities are identical for all reads r_n, we drop superscript n in this section for simplicity.

First, let us denote the probability of Z_j conditional on Z_i as follows. For $1 \le i \le j \le M$,

$$f_{11}(i, j) := \Pr(Z_j = 1|Z_i = 1) \tag{2.70}$$
$$f_{10}(i, j) := \Pr(Z_j = 0|Z_i = 1) \tag{2.71}$$
$$f_{01}(i, j) := \Pr(Z_j = 1|Z_i = 0) \tag{2.72}$$
$$f_{00}(i, j) := \Pr(Z_j = 0|Z_i = 0). \tag{2.73}$$

We can compute these quantities using DP as follows.

When $i = j$

$$f_{11}(i,j) := \Pr(Z_j = 1 | Z_i = 1) = 1 \tag{2.74}$$
$$f_{10}(i,j) := \Pr(Z_j = 0 | Z_i = 1) = 0 \tag{2.75}$$
$$f_{01}(i,j) := \Pr(Z_j = 1 | Z_i = 0) = 0 \tag{2.76}$$
$$f_{00}(i,j) := \Pr(Z_j = 0 | Z_i = 0) = 1. \tag{2.77}$$

When $i < j$

If two segments X_{j-1} and X_j are separated by an exon start site s_m (i.e., $j-1 = I(s_m)$):

$$f_{11}(i,j) := \Pr(Z_j = 1 | Z_i = 1) = \begin{cases} 1 & \text{if } i = j-1 \\ f_{11}(i,j-1) + f_{10}(i,j-1) p_m & \text{if } i < j-1, \end{cases} \tag{2.78}$$

because

$$\begin{aligned} &\Pr(Z_j = 1 | Z_i = 1) \\ &= \Pr(Z_j = 1, Z_{j-1} = 1 | Z_i = 1) + \Pr(Z_j = 1, Z_{j-1} = 0 | Z_i = 1) \\ &= \Pr(Z_j = 1 | Z_{j-1} = 1, Z_i = 1) \Pr(Z_{j-1} = 1 | Z_i = 1) + \Pr(Z_j = 1 | Z_{j-1} = 0, Z_i = 1) \Pr(Z_{j-1} = 0 | Z_i = 1) \\ &= \Pr(Z_j = 1 | Z_{j-1} = 1) \Pr(Z_{j-1} = 1 | Z_i = 1) + \Pr(Z_j = 1 | Z_{j-1} = 0) \Pr(Z_{j-1} = 0 | Z_i = 1) \\ &= f_{11}(i,j-1) + p_m f_{10}(i,j-1). \end{aligned} \tag{2.79}$$

Similarly,

$$f_{10}(i,j) := \Pr(Z_j = 0 | Z_i = 1) = \begin{cases} 0 & \text{if } i = j-1 \\ f_{10}(i,j-1)(1-p_m) & \text{if } i < j-1. \end{cases} \tag{2.80}$$

$$f_{01}(i,j) := \Pr(Z_j = 1 | Z_i = 0) = \begin{cases} p_m & \text{if } i = j-1 \\ f_{01}(i,j-1) + f_{00}(i,j-1) p_m & \text{if } i < j-1. \end{cases} \tag{2.81}$$

$$f_{00}(i,j) := \Pr(Z_j = 0 | Z_i = 0) = \begin{cases} 1-p_m & \text{if } i = j-1 \\ f_{00}(i,j-1)(1-p_m) & \text{if } i < j-1. \end{cases} \tag{2.82}$$

If two segments X_{j-1} and X_j are separated by an exon end site e_m (i.e., $j-1 = I(e_m)$):

$$f_{11}(i,j) := \Pr(Z_j = 1 | Z_i = 1) = \begin{cases} 1-q_m & \text{if } i = j-1 \\ f_{11}(i,j-1)(1-q_m) & \text{if } i < j-1. \end{cases} \tag{2.83}$$

$$f_{10}(i,j) := \Pr(Z_j = 0 | Z_i = 1) = \begin{cases} q_m & \text{if } i = j-1 \\ f_{10}(i,j-1) + f_{11}(i,j-1) q_m & \text{if } i < j-1. \end{cases} \tag{2.84}$$

$$f_{01}(i,j) := \Pr(Z_j = 1 | Z_i = 0) = \begin{cases} 0 & \text{if } i = j-1 \\ f_{01}(i,j-1)(1-q_m) & \text{if } i < j-1. \end{cases} \tag{2.85}$$

$$f_{00}(i,j) := \Pr(Z_j = 0 | Z_i = 0) = \begin{cases} 1 & \text{if } i = j-1 \\ f_{00}(i,j-1) + f_{01}(i,j-1) q_m & \text{if } i < j-1. \end{cases} \tag{2.86}$$

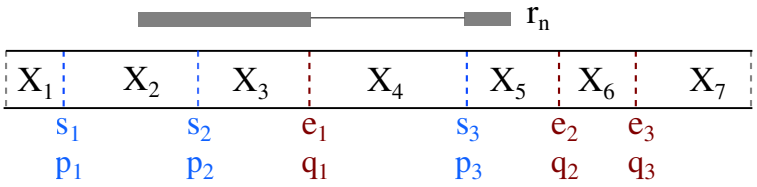


Figure 2.8: In this example, $s = z_{[2:5]}(1,1,0,1)$ and $w(s) = \Pr(Z_3 = 1, Z_4 = 0, Z_5 = 1 | Z_2 = 1) = 1 \cdot q_1 \cdot p_3$

Computation of $w(s) = \Pr(S = s | Z_{F(s)} = 1)$

The E-step in Section 2.2.4 used $\Pr(S_n = s | Z^n_{F(s)} = 1)$ to compute $\Pr(S_n = s | \Theta^l)$. These probabilities are also used in Sections 2.2.5, 2.2.5, 2.2.5, 2.2.5, and 2.2.5. Here we describe in detail how to compute them. As these quantities are identical for all reads r_n, we drop index n in this section for simplicity.

We use $w(s)$ to denote the probability of $S = s$ conditional on $X_{F(s)}$ is a part of an isoform. Let a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$. Due to the definition of S, that is the shortest subpath of T from which a read is derived, $o_a = 1, o_b = 1, F(s) = a$ and $L(s) = b$.

$$\begin{aligned} w(s) &= \Pr(S = s | Z_{F(s)} = 1) \\ &= \frac{\Pr(S = s)}{\Pr(Z_{F(s)} = 1)} \\ &= \frac{\Pr(Z_a = o_a, Z_{a+1} = o_{a+1}, \ldots, Z_b = o_b)}{\Pr(Z_a = 1)} \\ &= \Pr(Z_{a+1} = o_{a+1}, \ldots, Z_b = o_b | Z_a = 1). \end{aligned} \tag{2.87}$$

Moreover,

$$\begin{aligned} w(s) &= \Pr(Z_{a+1} = o_{a+1}, \ldots, Z_b = o_b | Z_a = 1) \\ &= \prod_{i=a}^{b-1} \Pr(Z_{i+1} = o_{i+1} | Z_i = o_i) \\ &= \prod_{s_m : a \le I(s_m) < b} p_m^{(1-o_{I(s_m)})(o_{[I(s_m)+1]})} (1-p_m)^{(1-o_{I(s_m)})(1-o_{[I(s_m)+1]})} \\ &\times \prod_{e_m : a \le I(e_m) < b} q_m^{(o_{I(e_m)})(1-o_{[I(e_m)+1]})} (1-q_m)^{(o_{I(e_m)})(o_{[I(e_m)+1]})}. \end{aligned} \tag{2.88}$$

In the example of Figure 2.8, $s = z_{[2:5]}(1,1,0,1)$. So $a = 2, b = 5, o_a = 1, o_b = 1, F(s) = 2$ and

$L(s) = 5$. Thus,

$$\begin{aligned}
w(s) &= \Pr(Z_3 = 1, Z_4 = 0, Z_5 = 1 | Z_2 = 1) \\
&= \Pr(Z_3 = 1 | Z_2 = 1) \Pr(Z_4 = 0 | Z_3 = 1) \Pr(Z_5 = 1 | Z_4 = 0) \\
&= 1 \cdot q_1 \cdot p_3 \\
&\text{or,} \\
&= \prod_{s_m : 2 \le I(s_m) < 5} p_m^{(1-o_{I(s_m)})(o_{[I(s_m)+1]})} (1-p_m)^{(1-o_{I(s_m)})(1-o_{[I(s_m)+1]})} \\
&\times \prod_{e_m : 2 \le I(e_m) < 5} q_m^{(o_{I(e_m)})(1-o_{[I(e_m)+1]})} (1-q_m)^{(o_{I(e_m)})(o_{[I(e_m)+1]})} \\
&= \prod_{s_m : s_2, s_3} p_m^{(1-o_{I(s_m)})(o_{[I(s_m)+1]})} (1-p_m)^{(1-o_{I(sm)})(1-o_{[I(sm)+1]})} \\
&\times \prod_{e_m : e_1} q_m^{(o_{I(em)})(1-o_{[I(em)+1]})} (1-q_m)^{(o_{I(em)})(o_{[I(em)+1]})} \\
&= p_2^{(1-1)(1)} (1-p_2)^{(1-1)(1-1)} p_3^{(1-0)(1)} (1-p_3)^{(1-0)(1-1)} \\
&\times q_1^{(1)(1-0)} (1-q_1)^{(1)(0)} \\
&= 1 \cdot p_3 \times q_1.
\end{aligned} \tag{2.89}$$

Computation of $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 1, S_n = s)$

The E-step in Section 2.2.4 used $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 1, S_n = s)$. Here, we describe how to compute these probabilities for the different cases when an exon start site s_m appears to the left, to the right, or within a subpath s. Figure 2.9 illustrates the different cases. As these quantities are identical for all reads r_n, we drop index n in this section for simplicity.

case 1: $I(s_m) < F(s)$

As shown in Figure 2.9, an exon start site s_m appears left side of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1, S = s) \\
&= \Pr(Z_1 = 0, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1, S = s) + \Pr(Z_1 = 1, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1, S = s) \\
&= \Pr(Z_1 = 0) \Pr(Z_{I(s_m)} = 0 | Z_1 = 0) \Pr(Z_{I(s_m)+1} = 1 | Z_{I(s_m)} = 0) \Pr(Z_{F(s)} = 1 | Z_{I(s_m)+1} = 1) \Pr(S = s | Z_{F(s)} = 1) \\
&+ \Pr(Z_1 = 1) \Pr(Z_{I(s_m)} = 0 | Z_1 = 1) \Pr(Z_{I(s_m)+1} = 1 | Z_{I(s_m)} = 0) \Pr(Z_{F(s)} = 1 | Z_{I(s_m)+1} = 1) \Pr(S = s | Z_{F(s)} = 1) \\
&= (1-\pi) f_{00}(1, I(s_m)) p_m f_{11}(I(s_m)+1, F(s)) w(s) + \pi f_{10}(1, I(s_m)) p_m f_{11}(I(s_m)+1, F(s)) w(s) \\
&= \left[(1-\pi) f_{00}(1, I(s_m)) + \pi f_{10}(1, I(s_m)) \right] \times p_m f_{11}(I(s_m)+1, F(s)) w(s),
\end{aligned} \tag{2.90}$$

where $f_{..}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

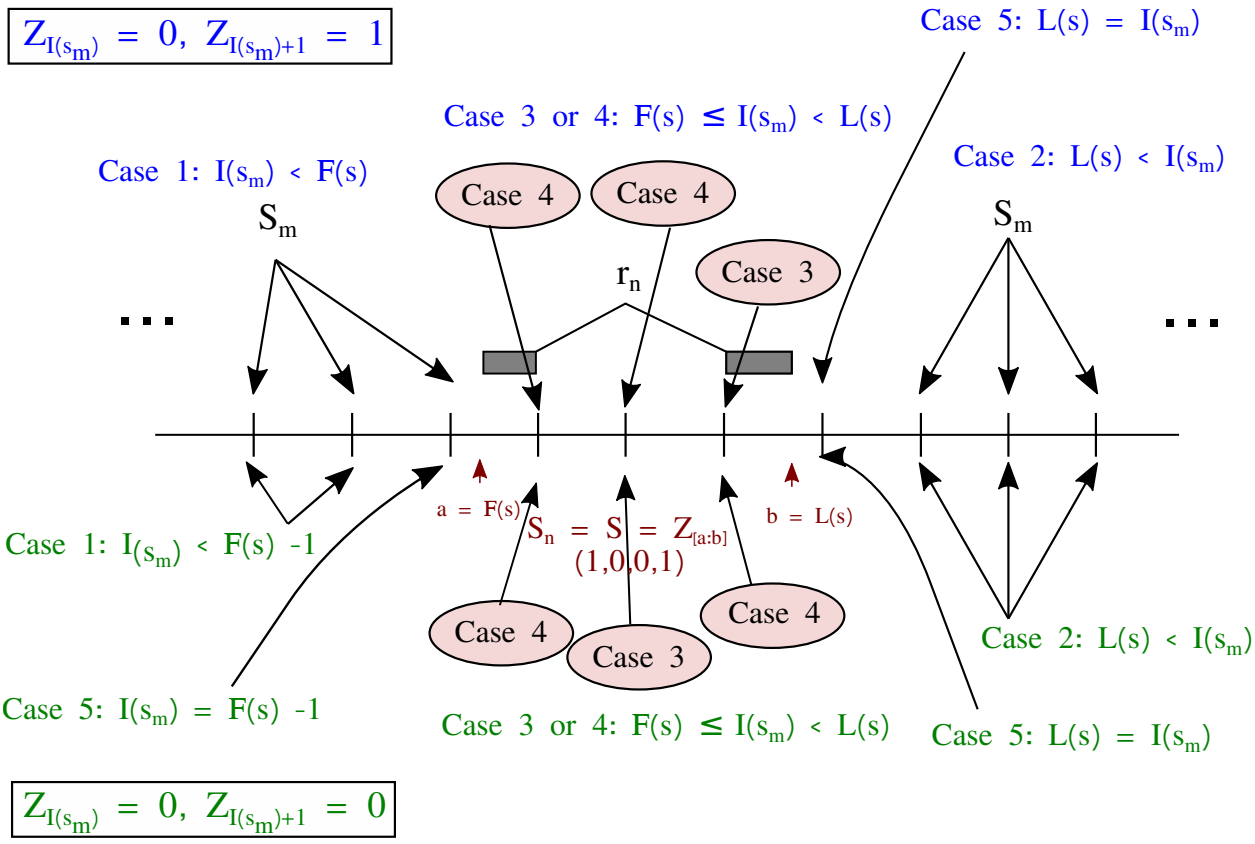


Figure 2.9: Visualization of the different cases considered for computing $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 1, S_n = s)$ in the upper part, and $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 0, S_n = s)$ in the lower part. Arrows from each case point to an exon start site or a set of exon start sites. a and b represent the indices of the first and last segments of the subpath $S_n = s$ from which read r_n is derived. In the upper part for $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 1, S_n = s)$, an exon start site s_m appears to the left of subpath s (case 1), to the right of s (cases 2 and 5), or within s (cases 3 and 4). We do not allow for cases 4 and 5 where $Z^n_{I(s_m)} = 0$ and $Z^n_{I(s_m)+1} = 1$ are not compatible with subpath s. In the lower part for $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 0, S_n = s)$, s_m appears to the left of s (cases 1 and 5), to the right of s (cases 2 and 5), or within s (cases 3 and 4). We do not allow for cases 4 and 5 where $Z^n_{I(s_m)} = 0$ and $Z^n_{I(s_m)+1} = 0$ are not compatible with subpath s.

case 2: $L(s) < I(s_m)$

As shown in Figure 2.9, s_m appears right side of the subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1) \\
&= \Pr(Z_1 = 0, S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1) + \Pr(Z_1 = 1, S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1) \\
&= \Pr(Z_1 = 0) \Pr(Z_{F(s)} = 1 | Z_1 = 0) \Pr(S = s | Z_{F(s)} = 1) \Pr(Z_{I(s_m)} = 0 | Z_{L(s)} = 1) \Pr(Z_{I(s_m)+1} = 1 | Z_{I(s_m)} = 0) \\
&+ \Pr(Z_1 = 1) \Pr(Z_{F(s)} = 1 | Z_1 = 1) \Pr(S = s | Z_{F(s)} = 1) \Pr(Z_{I(s_m)} = 0 | Z_{L(s)} = 1) \Pr(Z_{I(s_m)+1} = 1 | Z_{I(s_m)} = 0) \\
&= (1 - \pi) f_{01}(1, F(s)) w(s) f_{10}(L(s), I(s_m)) p_m + \pi f_{11}(1, F(s)) w(s) f_{10}(L(s), I(s_m)) p_m \\
&= \left[(1 - \pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s)) \right] \times w(s) f_{10}(L(s), I(s_m)) p_m,
\end{aligned} \tag{2.91}$$

where $f_{..}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 3: $F(s) \leq I(s_m) < L(s)$ **and** $(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1)$ **is a subset of** s

A subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$ can be represented by $(Z_a = o_a, Z_{a+1} = o_{a+1}, \ldots, Z_b = o_b)$. If the subpath s contains $(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1)$, then $(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1)$ is a subset of s (i.e., $(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1) \subset s$). As shown in Figure 2.9, s_m appears inside of the subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1) \\
&= \Pr(S = s) \\
&= \Pr(Z_1 = 0, S = s) + \Pr(Z_1 = 1, S = s) \\
&= \Pr(Z_1 = 0) \Pr(Z_{F(s)} = 1 | Z_1 = 0) \Pr(S = s | Z_{F(s)} = 1) + \Pr(Z_1 = 1) \Pr(Z_{F(s)} = 1 | Z_1 = 1) \Pr(S = s | Z_{F(s)} = 1) \\
&= (1 - \pi) f_{01}(1, F(s)) w(s) + \pi f_{11}(1, F(s)) w(s) \\
&= \left[(1 - \pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s)) \right] \times w(s),
\end{aligned} \tag{2.92}$$

where $f_{..}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 4: $F(s) \leq I(s_m) < L(s)$ **and** $(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1)$ **is not a subset of** s

In this case, $Z_{I(s_m)} = 0$ and $Z_{I(s_m)+1} = 1$ are not compatible with subpath s.

$$\Pr(S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1) = 0 \tag{2.93}$$

case 5: $I(s_m) = L(s)$

In this case, $Z_{I(s_m)} = 0$ and $Z_{I(s_m)+1} = 1$ are not compatible with subpath s.

$$\Pr(S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 1) = 0 \tag{2.94}$$

Computation of $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 0, S_n = s)$

The E-step in Section 2.2.4 used $\Pr(Z^n_{I(s_m)} = 0, Z^n_{I(s_m)+1} = 0, S_n = s)$. Here, we describe how to compute these probabilities for the different cases when an exon start site s_m appears to the left, to the right, or within a subpath s. Figure 2.9 illustrates the different cases. As these quantities are identical for all reads r_n, we drop index n in this section for simplicity.

case 1: $I(s_m) < F(s) - 1$

As shown in Figure 2.9, an exon start site s_m appears left side of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0, S = s) \\
&= \Pr(Z_1 = 0, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0, S = s) + \Pr(Z_1 = 1, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0, S = s) \\
&= \Pr(Z_1 = 0) \Pr(Z_{I(s_m)} = 0 | Z_1 = 0) \Pr(Z_{I(s_m)+1} = 0 | Z_{I(s_m)} = 0) \Pr(Z_{F(s)} = 1 | Z_{I(s_m)+1} = 0) \Pr(S = s | Z_{F(s)} = 1) \\
&+ \Pr(Z_1 = 1) \Pr(Z_{I(s_m)} = 0 | Z_1 = 1) \Pr(Z_{I(s_m)+1} = 0 | Z_{I(s_m)} = 0) \Pr(Z_{F(s)} = 1 | Z_{I(s_m)+1} = 0) \Pr(S = s | Z_{F(s)} = 1) \\
&= (1-\pi) f_{00}(1, I(s_m))(1-p_m) f_{01}(I(s_m)+1, F(s)) w(s) + \pi f_{10}(1, I(s_m))(1-p_m) f_{01}(I(s_m)+1, F(s)) w(s) \\
&= \left[(1-\pi) f_{00}(1, I(s_m)) + \pi f_{10}(1, I(s_m))\right] \times (1-p_m) f_{01}(I(s_m)+1, F(s)) w(s),
\end{aligned} \tag{2.95}$$

where $f_{\cdot\cdot}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 2: $L(s) < I(s_m)$

As shown in Figure 2.9, s_m appears right side of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0) \\
&= \Pr(Z_1 = 0, S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0) + \Pr(Z_1 = 1, S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0) \\
&= \Pr(Z_1 = 0) \Pr(Z_{F(s)} = 1 | Z_1 = 0) \Pr(S = s | Z_{F(s)} = 1) \Pr(Z_{I(s_m)} = 0 | Z_{L(s)} = 1) \Pr(Z_{I(s_m)+1} = 0 | Z_{I(s_m)} = 0) \\
&+ \Pr(Z_1 = 1) \Pr(Z_{F(s)} = 1 | Z_1 = 1) \Pr(S = s | Z_{F(s)} = 1) \Pr(Z_{I(s_m)} = 0 | Z_{L(s)} = 1) \Pr(Z_{I(s_m)+1} = 0 | Z_{I(s_m)} = 0) \\
&= (1-\pi) f_{01}(1, F(s)) w(s) f_{10}(L(s), I(s_m))(1-p_m) + \pi f_{11}(1, F(s)) w(s) f_{10}(L(s), I(s_m))(1-p_m) \\
&= \left[(1-\pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s))\right] \times w(s) f_{10}(L(s), I(s_m))(1-p_m),
\end{aligned} \tag{2.96}$$

where $f_{\cdot\cdot}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 3: $F(s) \leq I(s_m) < L(s)$ **and** $(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0)$ **is a subset of** s

As shown in Figure 2.9, s_m appears inside of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(s_m)} = 0, Z_{I(s_m)+0} = 1) \\
&= \Pr(S = s) \\
&= \Pr(Z_1 = 0, S = s) + \Pr(Z_1 = 1, S = s) \\
&= \Pr(Z_1 = 0) \Pr(Z_{F(s)} = 1 | Z_1 = 0) \Pr(S = s | Z_{F(s)} = 1) + \Pr(Z_1 = 1) \Pr(Z_{F(s)} = 1 | Z_1 = 1) \Pr(S = s | Z_{F(s)} = 1) \\
&= (1-\pi) f_{01}(1, F(s)) w(s) + \pi f_{11}(1, F(s)) w(s) \\
&= \left[(1-\pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s))\right] \times w(s),
\end{aligned} \tag{2.97}$$

where $f_{\cdot\cdot}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 4: $F(s) \leq I(s_m) < L(s)$ **and** $(Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0)$ **is not a subset of** s

In this case, $Z_{I(s_m)} = 0$ and $Z_{I(s_m)+1} = 0$ are not compatible with subpath s.

$$\Pr(s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0) = 0 \tag{2.98}$$

case 5: $I(s_m) = F(s) - 1$ **or** $I(s_m) = L(s)$

In this case, $Z_{I(s_m)} = 0$ and $Z_{I(s_m)+1} = 0$ are not compatible with subpath s.

$$\Pr(s, Z_{I(s_m)} = 0, Z_{I(s_m)+1} = 0) = 0 \tag{2.99}$$

Computation of $\Pr(Z^n_{I(e_m)} = 1, Z^n_{I(e_m)+1} = 1, S_n = s)$

The E-step in Section 2.2.4 used $\Pr(Z^n_{I(e_m)} = 1, Z^n_{I(e_m)+1} = 1, S_n = s)$. Here, we describe how to compute these probabilities for the different cases when an exon end site e_m appears to the left, to the right, or within a subpath s. Figure 2.10 illustrates the different cases. As these quantities are identical for all reads r_n, we drop index n in this section for simplicity.

case 1: $I(e_m) < F(s)$

As shown in Figure 2.10, an exon end site e_m appears left side of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1, S = s) \\
&= \Pr(Z_1 = 0, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1, S = s) + \Pr(Z_1 = 1, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1, S = s) \\
&= (1-\pi) f_{01}(1, I(e_m))(1-q_m) f_{11}(I(e_m)+1, F(s)) w(s) + \pi f_{11}(1, I(e_m))(1-q_m) f_{11}(I(e_m)+1, F(s)) w(s) \\
&= \left[(1-\pi) f_{01}(1, I(e_m)) + \pi f_{11}(1, I(e_m))\right] \times (1-q_m) f_{11}(I(e_m)+1, F(s)) w(s),
\end{aligned} \tag{2.100}$$

where $f_{..}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 2: $L(s) \leq I(s_m)$

As shown in Figure 2.10, e_m appears right side of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1) \\
&= \Pr(Z_1 = 0, S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1) + \Pr(Z_1 = 1, S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1) \\
&= (1-\pi) f_{01}(1, F(s)) w(s) f_{11}(L(s), I(e_m))(1-q_m) + \pi f_{11}(1, F(s)) w(s) f_{11}(L(s), I(e_m))(1-q_m) \\
&= \left[(1-\pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s))\right] \times w(s) f_{11}(L(s), I(e_m))(1-q_m),
\end{aligned} \tag{2.101}$$

where $f_{..}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 3: $F(s) \leq I(e_m) < L(s)$ **and** $(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1)$ **is a subset of** s

As shown in Figure 2.10, e_m appears inside of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1) \\
&= \Pr(S = s) \\
&= \Pr(Z_1 = 0, S = s) + \Pr(Z_1 = 1, S = s) \\
&= (1-\pi) f_{01}(1, F(s)) w(s) + \pi f_{11}(1, F(s)) w(s) \\
&= \left[(1-\pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s))\right] \times w(s),
\end{aligned} \tag{2.102}$$

where $f_{..}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

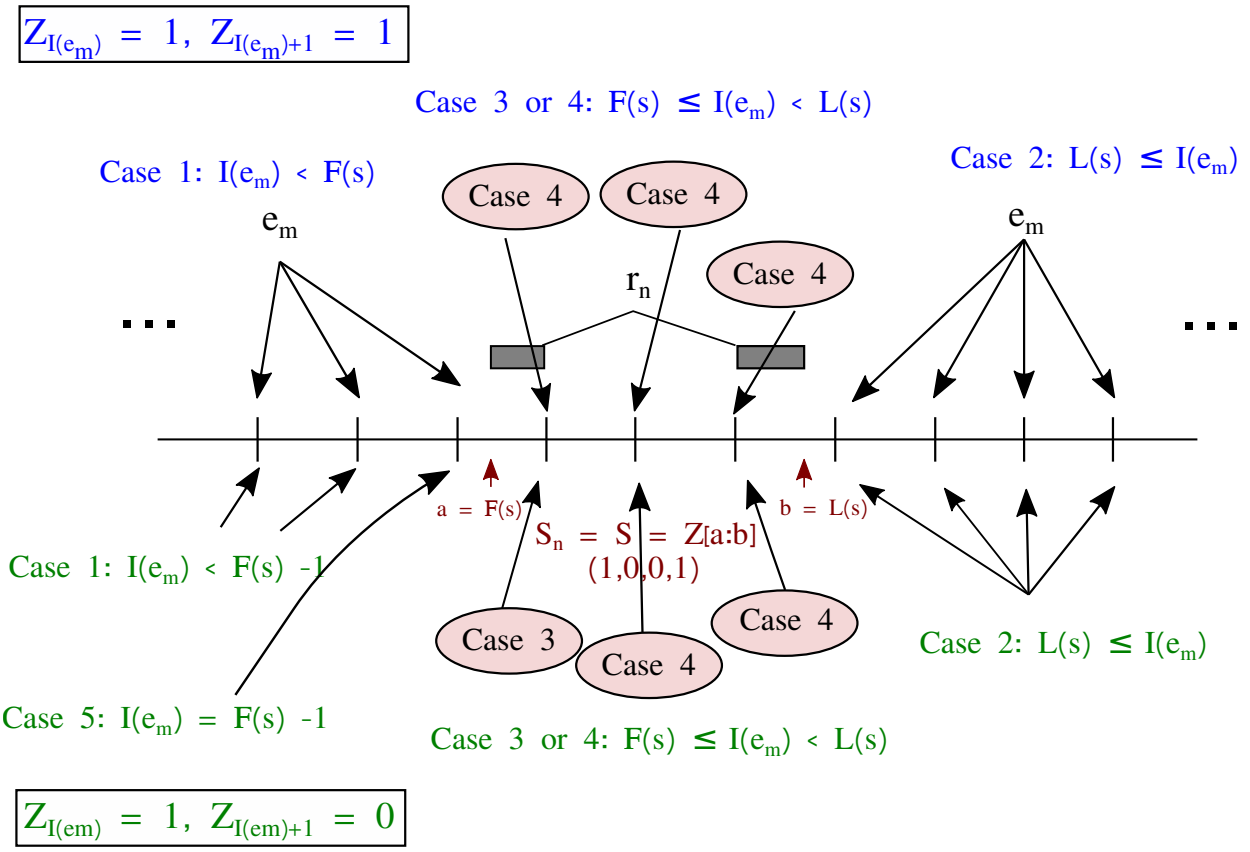


Figure 2.10: Visualization of the different cases considered for computing $\Pr(Z^n_{I(e_m)}=1, Z^n_{I(e_m)+1}=1, S_n=s)$ in the upper part, and $\Pr(Z^n_{I(e_m)}=1, Z^n_{I(e_m)+1}=0, S_n=s)$ in the lower part. Arrows from each case point to an exon end site or a set of exon end sites. a and b represent the indices of the first and last segments of the subpath $S_n=s$ from which read r_n is derived. In the upper part for $\Pr(Z^n_{I(e_m)}=1, Z^n_{I(e_m)+1}=1, S_n=s)$, an exon end site e_m appears to the left of subpath s (case 1), to the right of s (case 2), or within s (case 4). We do not allow for case 4 where $Z^n_{I(e_m)}=1$, $Z^n_{I(e_m)+1}=1$ are not compatible with subpath s. In the lower part for $\Pr(Z^n_{I(s_m)}=1, Z^n_{I(s_m)+1}=0, S_n=s)$, s_m appears to the left of s (cases 1 and 5), to the right of s (case 2), or within s (cases 3 and 4). We do not allow for cases 4 and 5 where $Z^n_{I(s_m)}=1$ and $Z^n_{I(s_m)+1}=0$ are not compatible with subpath s.

case 4: $F(s) \leq I(e_m) < L(s)$ **and** $(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 1)$ **is not a subset of** s

In this case, $Z_{I(e_m)} = 1$ and $Z_{I(e_m)+1} = 1$ are not compatible with subpath s.

$$\Pr(S = s, Z_{I(e_m)} = 1, Z_{I(s_m)+1} = 1) = 0 \tag{2.103}$$

Computation of $\Pr(Z^n_{I(e_m)} = 1, Z^n_{I(e_m)+1} = 0, S_n = s)$

The E-step in Section 2.2.4 used $\Pr(Z^n_{I(e_m)} = 1, Z^n_{I(e_m)+1} = 0, S_n = s)$. Here, we describe how to compute these probabilities for the different cases when an exon end site e_m appears to the left, to the right, or within a subpath s. Figure 2.10 illustrates the different cases. As these quantities are identical for all reads r_n, we drop index n in this section for simplicity.

case 1: $I(e_m) < F(s) - 1$

As shown in Figure 2.10, an exon end site e_m appears left side of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0, S = s) \\
&= \Pr(Z_1 = 0, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0, S = s) + \Pr(Z_1 = 1, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0, S = s) \\
&= (1-\pi) f_{01}(1, I(e_m)) q_m f_{01}(I(s_m)+1, F(s)) w(s) + \pi f_{11}(1, I(s_m)) q_m f_{01}(I(s_m)+1, F(s)) w(s) \\
&= \left[(1-\pi) f_{01}(1, I(e_m)) + \pi f_{11}(1, I(s_m))\right] \times q_m f_{01}(I(s_m)+1, F(s)) w(s),
\end{aligned} \tag{2.104}$$

where $f_{\cdot\cdot}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 2: $L(s) \leq I(e_m)$

As shown in Figure 2.10, e_m appears right side of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0) \\
&= \Pr(Z_1 = 0, S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0) + \Pr(Z_1 = 1, S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0) \\
&= (1-\pi) f_{01}(1, F(s)) w(s) f_{11}(L(s), I(s_m)) q_m + \pi f_{11}(1, F(s)) w(s) f_{11}(L(s), I(s_m)) q_m \\
&= \left[(1-\pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s))\right] \times w(s) f_{11}(L(s), I(s_m)) q_m,
\end{aligned} \tag{2.105}$$

where $f_{\cdot\cdot}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 3: $F(s) \leq I(e_m) < L(s)$ **and** $(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0)$ **is a subset of** s

As shown in Figure 2.10, e_m appears inside of a subpath $s = z_{[a:b]}(o_a, \ldots, o_b)$.

$$\begin{aligned}
&\Pr(S = s, Z_{I(e_m)} = 1, Z_{I(e_m)+0} = 1) \\
&= \Pr(S = s) \\
&= (1-\pi) f_{01}(1, F(s)) w(s) + \pi f_{11}(1, F(s)) w(s) \\
&= \left[(1-\pi) f_{01}(1, F(s)) + \pi f_{11}(1, F(s))\right] \times w(s),
\end{aligned} \tag{2.106}$$

where $f_{\cdot\cdot}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

case 4: $F(s) \leq I(e_m) < L(s)$ **and** $(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0)$ **is not a subset of** s

In this case, $Z_{I(e_m)} = 1$ and $Z_{I(e_m)+1} = 0$ are not compatible with subpath s.

$$\Pr(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0, S = s) = 0 \tag{2.107}$$

case 5: $I(e_m) = F(s) - 1$

In this case, $Z_{I(e_m)} = 1$ and $Z_{I(e_m)+1} = 0$ are not compatible with subpath s.

$$\Pr(Z_{I(e_m)} = 1, Z_{I(e_m)+1} = 0, S = s) = 0. \tag{2.108}$$

Computation of $\Pr(Z_1^n = 1, S_n = s)$ **and** $\Pr(Z_1^n = 0, S_n = s)$

The E-step in Section 2.2.4 used $\Pr(Z_1^n = 1, S_n = s)$ and $\Pr(Z_1^n = 0, S_n = s)$. Here, we describe how to compute these probabilities. As these quantities are identical for all reads r_n, we drop index n in this section for simplicity.

$$\begin{aligned} \Pr(Z_1 = 1, S = s) &= \Pr(Z_1 = 1) \Pr(Z_{F(s)} = 1 | Z_1 = 1) \Pr(S = s | Z_{F(s)} = 1) \\ &= \pi f_{11}(1, F(s)) w(s), \end{aligned} \tag{2.109}$$

and

$$\begin{aligned} \Pr(Z_1 = 0, S = s) &= \Pr(Z_1 = 0) \Pr(Z_{F(s)} = 1 | Z_1 = 0) \Pr(S = s | Z_{F(s)} = 1) \\ &= (1 - \pi) f_{01}(1, F(s)) w(s), \end{aligned} \tag{2.110}$$

where $f_{..}(i, j)$ and $w(s)$ can be computed as described in Sections 2.2.5 and 2.2.5.

2.3 Results

We assess the performance of McSplicer in comparison to existing state-of-the-art methods on both simulated and real RNA-seq data sets. Simulated data allow to compare estimates to a known ground truth of expressed transcripts and thus known quantities of alternative splicing events. On the other hand, simulated data cannot fully capture the complexity of data sets generated in real RNA-seq experiments. Note that exon start and end sites whose usage McSplicer estimates can correspond to splice sites but also to transcription start and end sites (see Section 2.1.2). In the following, however, we restrict the evaluation to the usage of splice sites since transcription start and end sites cannot be reliably estimated from short-read RNA-seq data alone.

We compare the performance of McSplicer to PSGInfer, SplAdder, MAJIQ, and StringTie. In Supplementary Section A.1 we provide details on software versions and command line arguments used. PSGInfer quantifies alternative splicing using a generative probabilistic model, an idea that also motivated the approach taken in McSplicer. SplAdder was used in a large-scale study (Kahles et al., 2018) to detect and quantify alternative splicing events in nearly 9,000 tumor RNA-seq samples. In a comparative benchmark analysis performed in Kahles et al. (2016) it showed a better performance than competing methods JuncBase (Brooks et al., 2011), rMATS (Shen et al., 2014), and SpliceGrapher (Rogers et al., 2012), from which, of course, general superiority cannot be concluded (Denti et al., 2018). Compared to SplAdder, which is limited to the detection of simple

types of splicing events, MAJIQ introduced a novel approach that additionally captures more complex transcript variations. MAJIQ was shown in a recent benchmark (Mehmood et al., 2019) to compare favorably to existing state-of-the-art methods and the authors demonstrated in Vaquero-Garcia et al. (2018) that MAJIQ also outperforms LeafCutter and rMATS.

StringTie, on the other hand, assembles and quantifies full-length transcripts from RNA-seq but was not specifically designed for the quantification of splice site usage. Nevertheless, splice site usage can be inferred from the abundance of the assembled transcripts and we include this approach as a baseline in our benchmark: In all experiments, McSplicer uses StringTie to construct the exon-intron structure in steps (B) and (C) of the workflow (Fig. 2.3), which potentially contains novel splice sites. In contrast to the inference of splice site usage from expressed full-length transcripts, however, McSplicer estimates the usage of the same set of splice sites using the EM algorithm described in the previous section.

Each method, however, uses a different set of parameters to quantify alternative splicing events. PSGInfer infers the weights of its constructed splice graph edges. SplAdder quantifies four canonical types of splicing events using the widely used *percent spliced in* (PSI) metric. PSI denotes the ratio between the number of reads supporting one outcome of the event (e.g the inclusion of an exon) over the number of reads directly supporting either of the two alternative outcomes. Similarly, MAJIQ computes the *percent selected index* (Ψ) for each splice junction involved in a *local splicing variation* (LSV), which denotes its fractional usage. To ensure a meaningful comparison of splice site usages in McSplicer to edge weights from PSGInfer, PSI from SplAdder and Ψ from MAJIQ, we only consider splice sites for which the meaning of these four quantities, if defined, coincide. These *comparable* splice sites are obtained from alternative splicing events between two expressed transcripts such that all remaining transcripts expressed by a gene consistently support one of the two possible outcomes of the event. Note that comparable splice sites are defined based on transcripts expressed in a given sample.

Comparable splice sites

Let $s_1, s_2, \ldots, s_{M_G}$ denote the splice sites and transcription start and end sites of a gene G, ordered by their genomic coordinates. Consistent with Foissac and Sammeth (2007), we define *alternative splicing events* for pairs of expressed transcripts t_1, t_2 as maximal sequences $s_i, \ldots, s_j$ of alternative splice sites, i.e. splice sites that are used by t_1 or by t_2, but not by both. To distinguish the outcome of alternative splicing from the outcome of alternative transcription initiation or termination, we additionally require that s_{i-1} and s_{j+1} denote common donor and acceptor sites, respectively. If every transcript expressed by G is consistent with t_1 or t_2 in its use of $s_{i-1}, s_i, \ldots, s_{j+1}$, we call the alternative splice sites $s_i, \ldots, s_j$ *comparable*. Note that the definition of comparable splice sites is invariant with respect to the choice of t_1 and t_2 among expressed transcripts of G.

For comparable splice sites of simple events, the four different parameters, i.e. splice site usage, edge weights, PSI and Ψ, equally reflect the relative abundance of transcripts expressed by a given gene that use the splice site, or equivalently contain the corresponding exon. Analogously, Ψ, edge weight, and splice site usage are equivalent for comparable splice sites of complex events. We will therefore consistently refer to these different parameters in the following as splice site usage. From StringTie assemblies of full-length transcripts, estimates of splice site usage can directly be obtained from the relative abundance of transcripts using a given splice site. For an illustrative example of comparable and non-comparable splice sites see Fig. 2.11.

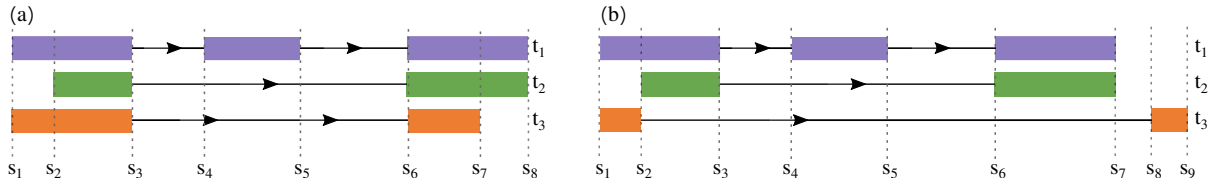


Figure 2.11: (a) An example of an exon skipping event involving *comparable* splice sites. The splice sites s_4 and s_5 are used exclusively by t_1, but splice sites s_3 and s_6 are common donor and acceptor sites to all three transcripts t_1, t_2, and t_3. (b) An example of an exon skipping event with non-comparable splice sites. The splice sites s_4 and s_5 are used exclusively by t_1. The splice sites s_3 and s_6 denote the common donor and acceptor sites of t_1 and t_2, Transcript t_3, however, is inconsistent with both t_1 and t_2 in its use of splice sites $s_3 \ldots s_6$.

2.3.1 McSplicer more accurately infers splice site usage than competing methods

In this section, we assess the performance of McSplicer on RNA-seq data sets simulated as described in Section 2.1.4. All methods but PSGInfer were provided the same set of reads aligned using STAR (allowing mismatches and indels). PSGInfer only accepted unaligned reads which were internally mapped using Bowtie (Langmead et al., 2009). We distinguish splice sites by the type of event they are part of, including exon skipping, intron retention, alternative $3'$ and $5'$ splice sites, and complex events that cannot be assigned to one of the canonical types. The events are labeled by Astalavista (Foissac and Sammeth, 2007) through a pairwise structural comparison of all transcript species expressed in our ground truth transcriptome (see Fig. 2.2 and Fig. 2.12).

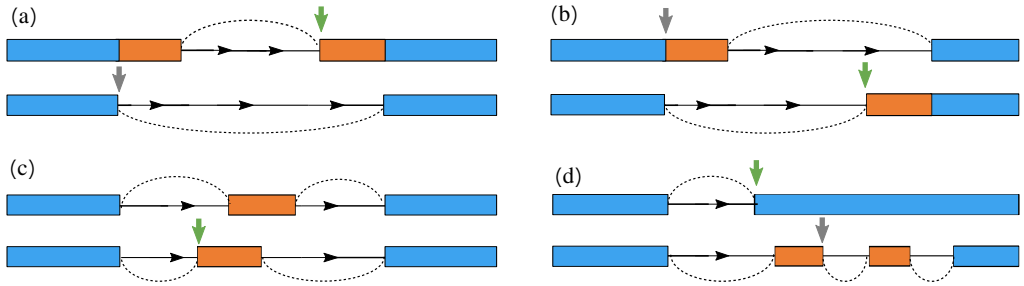


Figure 2.12: Examples of complex patterns of alternative splicing Foissac and Sammeth (2007). Green and grey arrows highlight the corresponding varying donor and acceptor splice sites, respectively. These are illustrative examples of the varying non-redundant splice sites which we consider in our benchmark.

The number of variable splice sites (i.e. $0 < \text{usage} < 1$) in our simulated data set, and the number of comparable splice sites among them ($\sim 36\%$), with corresponding event types defined by Astalavista are listed in Table 2.2. It also lists the total number of (comparable) splice sites per type reported by all four methods. While McSplicer will quantify the usage of all splice sites except those missed by StringTie in step (B) in Fig. 2.3, competing methods report only events that satisfy an adjustable confidence threshold (SplAdder) or are considered reliable according to internal filters (MAJIQ). As a result, both MAJIQ's and SplAdder's accuracy is evaluated on a smaller, presumably more confidently estimated set of events (Table 2.2) and are otherwise not penalized for missing events. MAJIQ estimates two parameters that correspond to the relative usage of a skipped exon, one based on the intron connecting it to the upstream exon, and one based on the downstream exon (Fig. 2.13). Here, we compare the performance to the latter one, which we observed to be slightly more accurate. The former is reported in Fig. 2.14.

	Exon skipping	Alt. acceptor	Alt. donor	Intron retention	Complex	Tota
AStalavista (all)	1740	544	295	318	1206	4103
AStalavista (comparable)	475	229	129	134	508	1475
MAJIQ (comparable)	371	106	81	89	429	1076
SplAdder (comparable)	366	150	87	25	-	628
PSGInfer (comparable)	330	127	56	128	88	729
StringTie (comparable)	455	209	120	127	487	1398
McSplicer (comparable)	455	209	120	127	487	1398
McSplicer (non-comp.)	1070	292	153	180	502	2197

Table 2.2: The first row shows the total number of variable splice sites (i.e., comparable and non-comparable), while the second row provides the number of comparable splice sites among them. The values in the first two rows are obtained from ground truth transcript expressions and classified by type as labeled by AStalavista. Each simple event contains by definition one variable splice site whose usage uniquely quantifies the event (see Figure 2.2), while for complex events we consider one or two variable splice sites that are comparable (see Figure 2.12). The following rows show the number of variable splice sites classified by event type as quantified by each of the five methods in the simulated RNA-seq data set with 50 million reads. For McSplicer we additionally provide the number of non-comparable sites quantified. Note that McSplicer estimates the usage of all splice sites reported by StringTie. StringTie correctly identifies approximately 96% of all splice sites in our benchmark (computed from the values above) and reports few false splice sites (precision $\approx 96\%$).

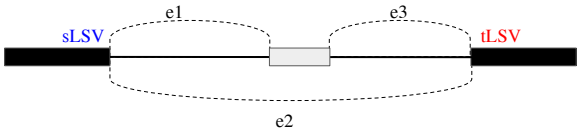


Figure 2.13: MAJIQ computes the percent selected index (ψ) for each junction involved in a local splicing variation (LSV) which denotes its fractional usage. An exon skipping event can be inferred either from the estimated ψ value of edge e_1 connecting source LSV (*sLSV*) to the cassette exon, or edge e_3 connecting the cassette exon to the target LSV (*tLSV*). We notice that the estimated usage $E[PSI(e_3)]$ tends to be slightly more accurate than $E[PSI(e_1)]$.

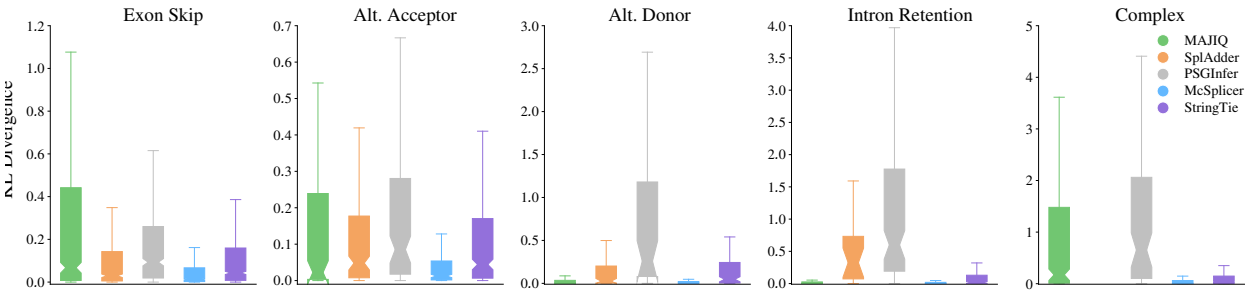


Figure 2.14: Accuracy of McSplicer and competing methods in quantifying the usage of variable splice sites from 50 million simulated RNA-seq reads. For MAJIQ, here we consider the estimated ψ value of the edge incident to the source LSV. See Fig. 2.13 for an illustration.

Fig. 2.15 compares the accuracy of splice site usages inferred by McSplicer and competing methods from 50 million reads on four canonical types of events as well as on complex events. For each method, only events reported and quantified by that method are considered. Fig. 2.16

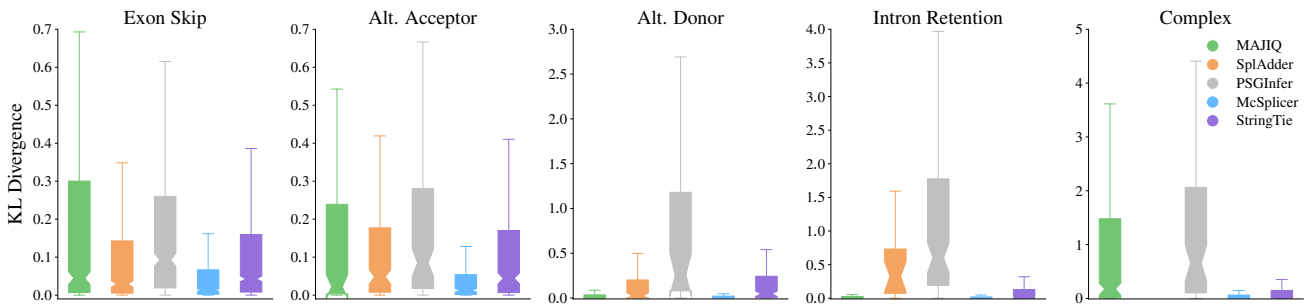


Figure 2.15: Accuracy of McSplicer and competing methods in quantifying the usage of variable splice sites from 50 million simulated RNA-seq reads. For each method, only splice sites in events that the method reports and quantifies are considered. SplAdder is limited to the quantification of simple AS events.

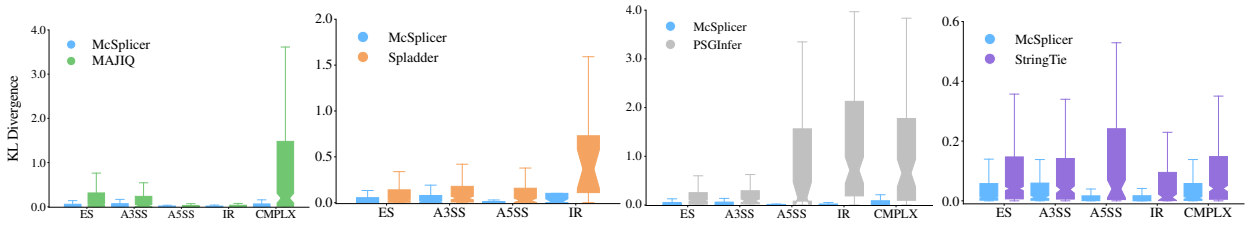


Figure 2.16: Accuracy of McSplicer and competing methods in quantifying the usage of variable splice sites from 50 million simulated RNA-seq reads. Events that McSplicer and competing methods have pairwise in common are considered. SplAdder is limited to the quantification of simple AS events.

shows consistent results when considering events that McSplicer and competing methods have pairwise in common. Across all types of events, McSplicer infers splice site usages more accurately than competing methods. The accuracy of splice site usage inferred by McSplicer is not affected by the complexity of the event, whereas MAJIQ's estimates are substantially less accurate for complex events. SplAdder is restricted to the quantification of simple events. As originally reported by the authors in Kahles et al. (2016), SplAdder quantifies intron retentions less accurately than other simple types of events. Other methods, including McSplicer, perform well on this type of event, which plays an important role for cell development in mammals (Braunschweig et al., 2014) and is a source of neoepitopes in cancer (Smart et al., 2018). We note that different read alignments used in PSGInfer cannot be excluded as a potential contributor to its overall low accuracy. Compared to baseline splice site usage extracted from StringTie transcript assemblies, McSplicer utilizes StringTie's transcript models to substantially refine the quantification of local splicing variation. We would like to point out, however, that StringTie was designed to assemble full-length transcripts. The comparison to StringTie merely highlights the necessity of additional computations to obtain more accurate estimates of splice site usage. Similar results were obtained on data sets comprising 20 million and 75 million reads (see Figs. 2.18 and 2.19). Furthermore, we demonstrate in Fig. 2.17 that McSplicer also achieves accurate estimates on the more challenging set of non-comparable splice sites. While KL divergences are slightly higher than on comparable splice sites, its estimates remain more accurate compared to competing methods that are evaluated only on a subset of comparable splice sites.

Fig. 2.20 shows running times of all methods on the three simulated data sets. The splicing model underlying McSplicer allows a much faster estimation of parameters than PSGInfer ($\sim$1 h

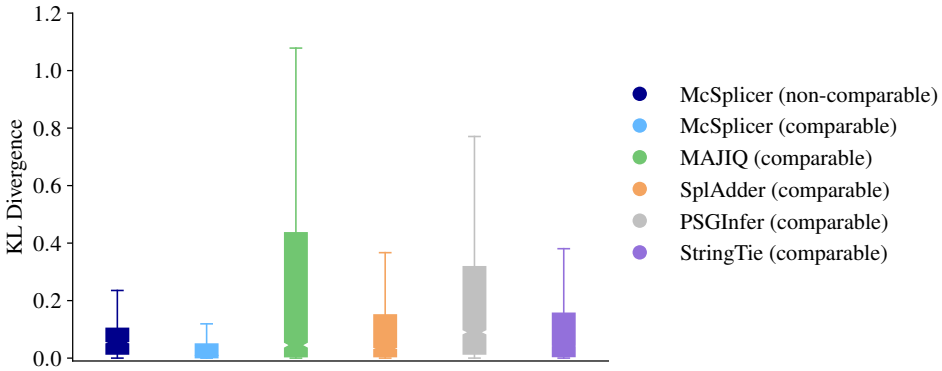


Figure 2.17: Accuracy of McSplicer and competing methods in quantifying the usage of comparable vs. non-comparable splice sites from 50 million simulated RNA-seq reads. For each method we report KL divergences on comparable splice sites over all types of events. For McSplicer we additionally show KL divergences for all non-comparable splice sites. Note that KL divergences reported for SplAdder do not include complex events, on which MAJIQ and PSGInfer obtained substantially less accurate estimates, see Table 2.2. In contrast to comparable splice sites that are included or excluded in only one unique way across all expressed transcripts, non-comparable splice sites may be overlapped by an arbitrary number of transcripts with varying exon-intron structure.

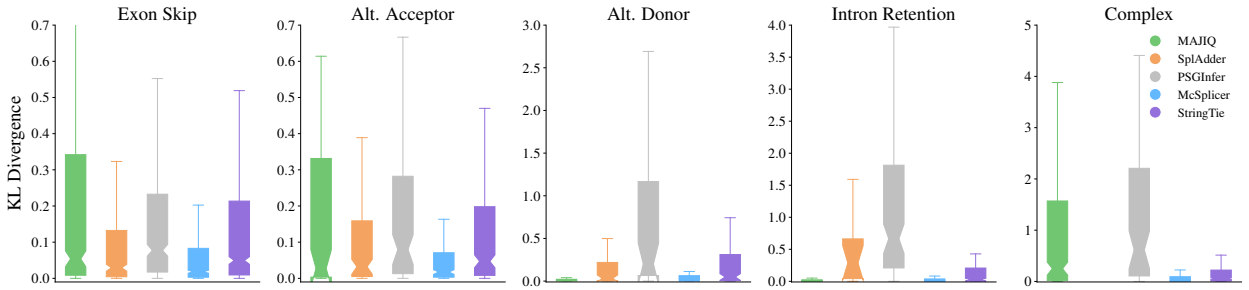


Figure 2.18: Accuracy of McSplicer and competing methods in quantifying the usage of variable splice sites from 20 million simulated RNA-seq reads. For each method, only splice sites in events that the method reports and quantifies are considered. SplAdder is limited to the quantification of simple events.

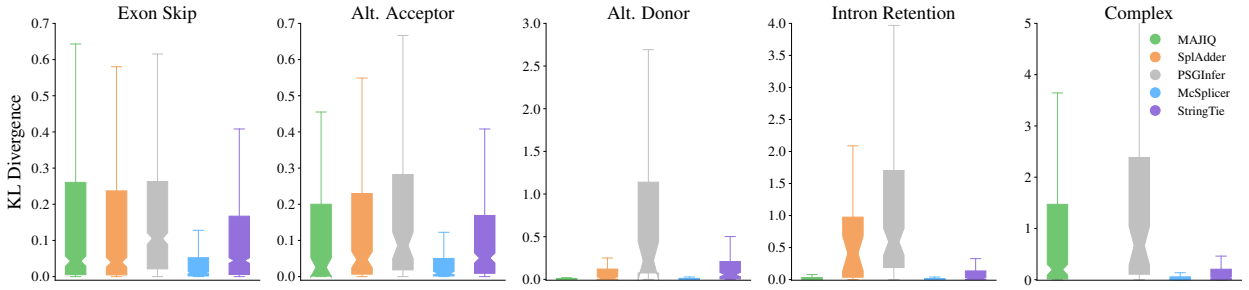
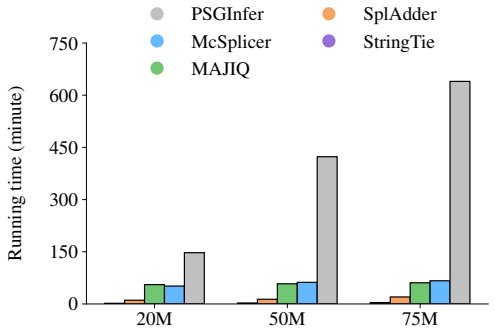


Figure 2.19: Accuracy of McSplicer and competing methods in quantifying the usage of variable splice sites from 75 million simulated RNA-seq reads. For each method, only splice sites in events that the method reports and quantifies are considered. SplAdder is limited to the quantification of simple events.

vs. 7 h for 50 million reads), the only other method that is based on a probabilistic model of the splicing process. MAJIQ similarly required around 1 h. As expected, the computation of read count ratios makes SplAdder the fastest method among direct competitors (< 14 min). StringTie is by far the fastest method (< 3 min), albeit solving a different task. Peak memory usage was below 3 GB for all methods except PSGInfer, which however included as the only method the read mapping step (Fig. 2.21).



	20M	50M	75M
PSGInfer	147.37	423.17	639.72
McSplicer	51.47	62.11	66.63
MAJIQ	55.62	58.02	60.87
SplAdder	10.77	13.50	20.14
StringTie	1.93	2.77	4.06

Figure 2.20: Running times in minutes of PSGInfer, McSplicer, MAJIQ, SplAdder and StringTie on the 20, 50, and 75 million simulated RNA-seq reads data sets. The running time reported for McSplicer includes the time needed to partition genes into non-overlapping segments and to count reads that map to the same sequence of segments (*signature counts*, see Methods) The running times were measured on an Intel Xeon CPU @2.30GHz with 320 GB memory. McSplicer, MAJIQ, and SplAdder were run in single-thread mode, while PSGInfer was run with 72 threads to speed up computation.

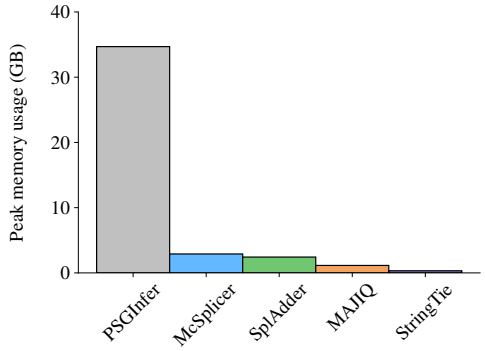


Figure 2.21: Peak memory usage measured for all methods on the largest simulated RNA-seq data set with 75 million reads. Peak memory usages were 34.69 GB for PSGInfer, 2.90 GB for McSplicer, 2.43 GB for SplAdder, 1.14 GB for MAJIQ, and 0.33 GB for StringTie. Note that memory usage of PSGInfer includes the read mapping step using Bowtie (Langmead et al., 2009) which could not be separated from PSGInfer's inference algorithm called by a single command *psg_infer_frequencies*. All other methods exclude read mapping.

2.3.2 McSplicer leverages all reads mapped to a gene

McSplicer makes use of all reads mapped to a given gene to simultaneously infer parameters in the McSplicer model, while other methods except PSGInfer typically use only reads that directly support their parameters. To quantify the contribution of the simultaneous inference in McSplicer to improve the accuracy of estimators, we estimate one splice site usage parameter at a time using only reads directly supporting the parameter. Similar to the calculation of the traditional PSI metric, we remove for each event with comparable splice sites all reads that do not overlap any of the event's exons, and run and evaluate McSplicer on the resulting restricted instance as described in the previous section. Fig. 2.22 confirms that McSplicer profits enormously from transcriptional evidence that lies outside of the local splicing event. Across all types of events, McSplicer estimates splice site usage less accurately when reads that do not overlap an event are removed.

2.3.3 McSplicer estimates agree with Spike-In RNA Variants

To evaluate the performance of McSplicer under the added complexity imposed by data derived from a real RNA-seq experiment, we used spike-in controls that were previously added to human monocyte-derived macrophages from five different donors (Hoss et al., 2019). The Spike-In RNA Variants (SIRV) (Paul et al., 2016) comprise 69 synthetic RNA molecules that were added in known relative concentrations before library preparation. Mimicking the complexity of 7 human model genes, between 6 and 18 artificial transcripts per gene vary in different types of alternative splicing, transcription start- and end-sites, or are transcribed from overlapping genes, or the antisense strand. The concentration ratios between different SIRV isoforms span a range of more than two orders of magnitude. For each donor sample, including artificial SIRV isoforms, Hoss et al. (2019) sequenced 200 million paired-end reads of 2×125bp length. McSplicer considers both mates independently as input reads R_n (see Section 2.1.3).

Leveraging the artificial reference genome (SIRVome) and the known relative mixing ratios of SIRV isoforms, we derive ground truth splice site usages (see Equation 2.12). Again, we obtain event labels from Astalavista, which comprise 26 variable splice sites in simple events and 12 in complex events. In this experiment, we do not restrict the evaluation to comparable splice

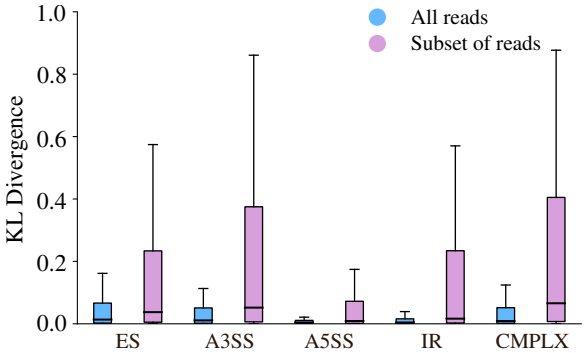


Figure 2.22: McSplicer leverages all RNA-seq reads mapped to a gene to improve the accuracy of splice site usage estimates. On the data set with 50 million simulated reads, McSplicer achieves lower KL divergence from true splice site usages when considering all reads mapped to a gene locus at once (blue) compared to using only reads that overlap any of the event's exons (pink). ES denotes exon skipping, A3SS alternative 3' splice site, A5SS alternative 5' splice site, IR intron retention, and CMPLX complex events.

sites but include all variable sites since competing methods report too few events to be compared quantitatively (see below). Fig. 2.23 compares splice site usages as estimated by McSplicer to the true usages in one of the five samples (donor 5). A Spearman's rank correlation coefficient of $\rho = 0.798$ indicates a good agreement between estimated and true usages. We obtain similar results on the remaining four samples (Fig. 2.24).

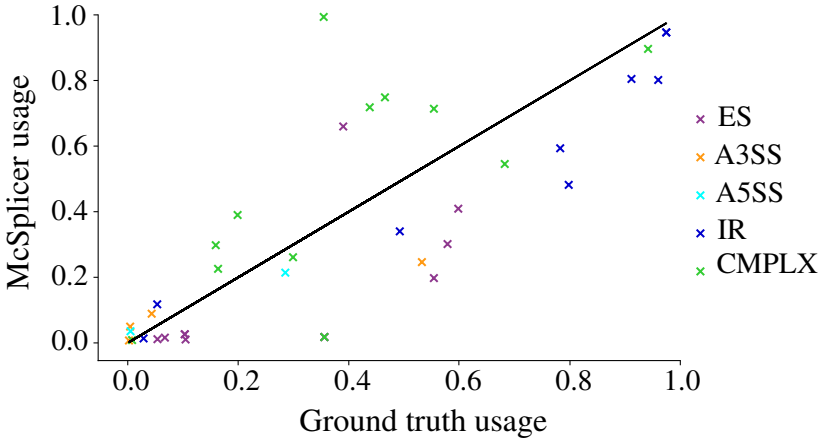


Figure 2.23: McSplicer results on spike-in RNA variants (SIRV), donor sample 5. Ground truth splice site usages computed from known mixing ratios of SIRV isoforms are compared to usages estimated by McSplicer. Out of 38 variable splice sites, 26 belong to simple events and 12 belong to complex events. ES denotes exon skipping, A3SS alternative 3' splice site, A5SS alternative 5' splice site, IR intron retention, and CMPLX complex events.

SplAdder and MAJIQ only report between 6 and 12 among all 38 true events, too few to allow for a meaningful quantification of agreement between estimated and true PSI and ψ values. Figs. 2.25 and 2.26 show the corresponding scatter plots for PSI and ψ values estimated by SplAdder and MAJIQ, respectively. PSGInfer failed to run on all five donor samples for unknown reasons.

2.3.4 Quantifying the effect of cryptic splice site mutations in patients with autism spectrum disorder

In this section, we illustrate the utility of splice site usages computed by McSplicer in interpreting the potentially complex effect of genetic variants on RNA splicing. In Jaganathan et al. (2019), the authors use a deep neural network to identify non-coding genetic variants that disrupt mRNA splicing. They identified a set of high-confidence *de novo* mutations predicted to disrupt splicing in individuals with intellectual disability and individuals with autism spectrum disorders (ASD). To validate them, the study included RNA-seq experiments (270-388 million 150bp reads per sample) of peripheral blood-derived lymphoblastoid cell lines from 36 individuals with ASD. Based on the presence of reads spanning the corresponding splice junction, the authors validate 21 aberrant splicing events associated with the predicted *de novo* mutations. Each of the splicing events was uniquely observed in one individual.

In Jaganathan et al. (2019), the authors point out that computing the effects size of splicing mutations based on a pre-selected set of incident splice junctions likely underestimates the true effect size since, among other shortcomings, not all isoform changes are taken into account. In contrast, McSplicer's model of splice site usage does not depend on an ad hoc selection of specific junctions or AS units but naturally captures simultaneous changes in expression of multiple isoforms expressed by a gene. We therefore utilized McSplicer to quantify the effect size of the validated

de novo mutations on splice sites in ASD patients. We excluded 11 aberrant splicing events where only 1 or 2 spliced reads supported the novel splice site or junction. For each *de novo* mutation and the corresponding aberrant splicing event, we used McSplicer to estimate splice site usage and to compute 95% bootstrapping confidence intervals for the individual harboring the variant and a control individual with similar sequencing depth. For all 10 aberrant splicing events, we observe significantly different splice site usages (i.e., the two confidence intervals do not overlap) between mutated and control ASD individuals (Table 2.3). Fig. 2.27 provides three illustrative examples. For gene *ENOPH1*, McSplicer estimates a decrease in usage of the acceptor site directly affected by the variant, consistent with the increased skipping of the corresponding exon that can be observed in the Sashimi plot. In gene *CORO1B*, a novel donor site is used exclusively in the individual with the variant, identified and quantified with non-zero usage by McSplicer. For gene *PCSK7*, McSplicer estimates a decrease in usage of the affected donor sites, consistent with the retention of the downstream intron.

Gene name	chr	Splice Site	Mutated	Control	Effect size	Event type
BCL7B	7	72966572	0.786 (0.784,0.792)	0.956 (0.953,0.960)	-1.06	ES
ENOPH1	4	83378068	0.624 (0.622, 0.628)	0.991 (0.991,0.993)	-0.20	ES
YME1L1	10	27431414	0.353 (0.349,0.356)	0.81 (0.810,0.813)	-0.36	ES
PPP4R2	3	73112824	0.463 (0.459,0.466)	0.951 (0.950,0.955)	-0.32	ES
TMBIM6	12	50153004	0.887 (0.887,0.889)	0.945 (0.948,0.949)	-0.05	ES
IDUA	4	997837	0.209 (0.208,0.211)	0.0	∞	Novel A5SS
CORO1B	11	67208804	0.054 (0.051,0.055)	0.0	∞	Novel A5SS
SHPRH	6	146266702	0.546 (0.529,0.582)	0.0	∞	Novel IR
PCSK7	11	117098932	0.67 (0.631,0.776)	0.969 (0.967,0.971)	-0.16	Novel IR
ELOVL1	1	43829994	0.200 (0.200,0.215)	0.0	∞	Novel IR

Table 2.3: McSplicer splice site usage estimates on mutated and control Autism samples with 95% bootstrapping confidence intervals shown in parentheses. We compute the effect size using the difference in the estimated splice site usages between mutated and control samples in log scale. There is no RNA-seq read evidence supporting the novel splice sites for the control samples in genes IDUA, CORO1B, SHPRH, and ELOVL1, hence we report the usage estimate as 0 and the effect size for these genes as ∞.

2.4 Conclusion

We have introduced McSplicer, a novel method that estimates the usage of exon start and end sites, and in particular the usage of splice sites across expressed transcripts. Rather than attempting to reconstruct expressed transcripts, McSplicer is based on a simplified probabilistic splicing model that has generated the set of expressed transcripts. It is not restricted to a pre-defined class of alternative splicing events or units but our probabilistic model is able to describe arbitrarily complex types of splicing patterns based on few, easy to interpret, parameters. We estimate these parameters, i.e. splice site usages, using all read data at once and demonstrate in simulation experiments that this yields more accurate estimates compared to other methods that use only reads directly supporting their parameters. Through its integration with transcript assembly methods such as StringTie, McSplicer quantifies the usage of annotated as well as novel splice sites.

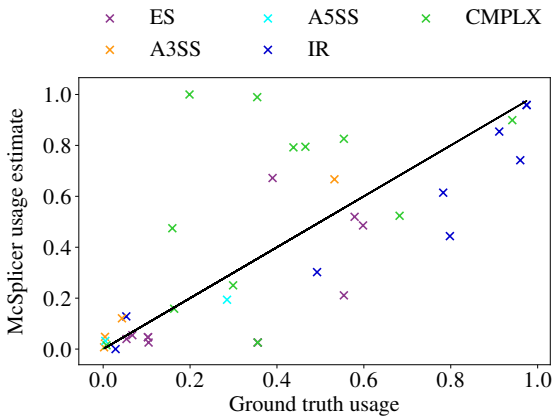
Our model for relative transcript abundance assumes the Markovian property across indicators (Z_i) for whether a segment is transcribed. This assumption allows for an efficient algorithm to estimate parameters of the model, but it potentially limits the ability of our method to model

longer range dependencies such as between the recognition of 5' and 3' splice sites or between the removal of introns within transcripts. If true dependencies are longer than our model can describe, the individual estimators for splice site usages may still be accurate, but we expect transcript frequencies implied by our model to be less accurate (LeGault and Dewey, 2013). One way to model longer range dependencies is to use higher order Markov chains as long as the data provide sufficient information to estimate these dependencies.

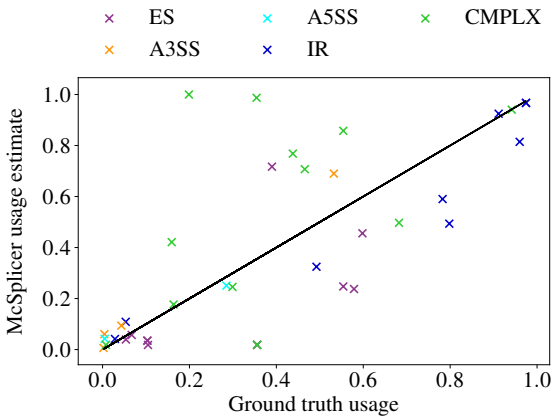
The splice site usages computed by McSplicer can be leveraged in various types of downstream analyses, such as the statistical comparison of splice site usage between different conditions (Li et al., 2018), the quantification of various types of splicing events, the identification of subgroups of samples that show similar splicing patterns (i.e., unsupervised clustering (Ntranos et al., 2016)), or the discrimination between alternatively spliced and constitutive exons (Patrick et al., 2013).

We have used McSplicer to quantify the effect size of splicing mutations in ASD patients. In this context, splice site usage as computed by McSplicer can be considered analogous to the "strength" of a splice site predicted by methods such as SplicePort (Dogan et al., 2007) from sequence-based features. Point mutations in the consensus splice site sequence can affect the strength of a splice site and result in the skipping of the exon or the activation of cryptic splice sites. In fact, a single nucleotide substitution might produce multiple (erroneous) splicing isoforms at the same time, as has been observed, for example, for specific mutations in patients with cystic fibrosis (3 isoforms) (Ramalho et al., 2003) and X-linked spondyloepiphyseal dysplasia tarda (7 isoforms) (Xiong et al., 2009). McSplicer does not attempt to reconstruct every single aberrant isoform, but similar to a weakening (strengthening) of a splice site as predicted from sequence alterations by, e.g., the Shapiro splice site probability score (Shapiro and Senapathy, 1987), the effect of a mutation will be reflected in a reduced or increased usage of the corresponding splice site estimated from RNA-seq reads.

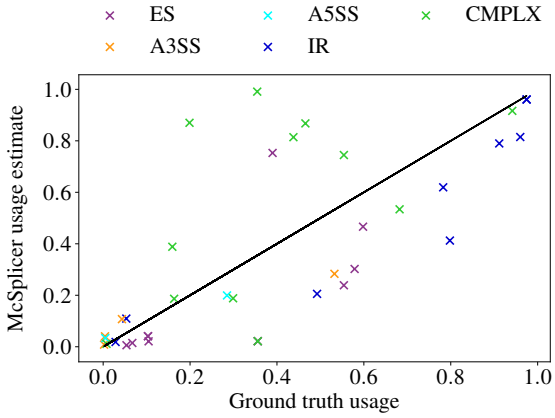
The procedure we applied to compute the effect size of splicing mutations in our analysis of ASD patients data does not use the full data from multiple individuals and fails to consider variability among individuals, possibly leading to an increased number of false positives. Methods that model differences in splice site usages between individuals from multiple groups and exploit the variability among them should perform better in estimating effect size and quantifying their uncertainty.



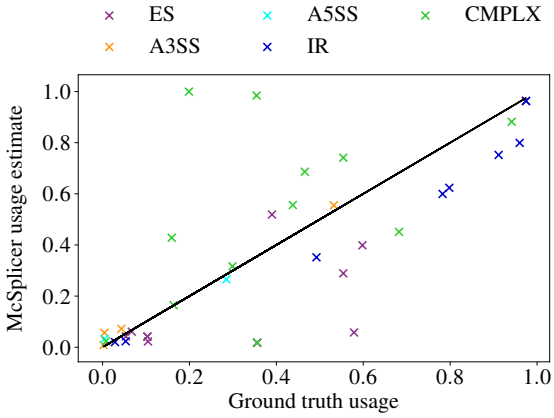
(a) Donor 1. Spearman's $\rho = 0.774$.



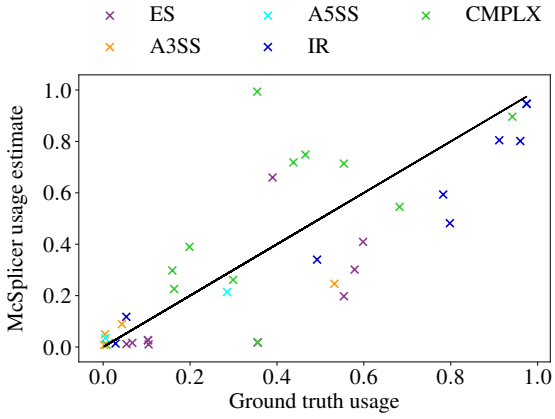
(b) Donor 2. Spearman's $\rho = 0.769$.



(c) Donor 3. Spearman's $\rho = 0.774$.

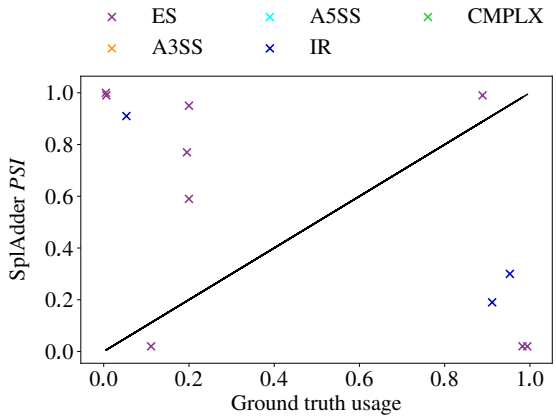


(d) Donor 4. Spearman's $\rho = 0.782$.

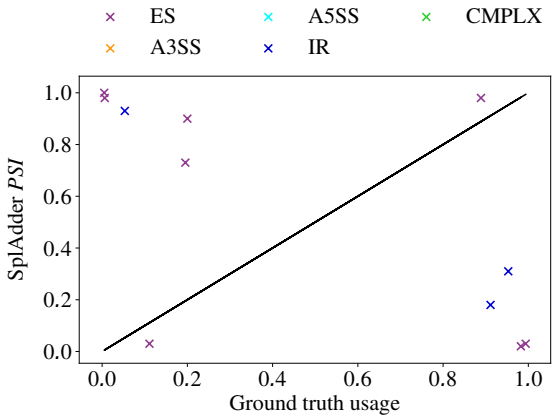


(e) Donor 5. Spearman's $\rho = 0.798$.

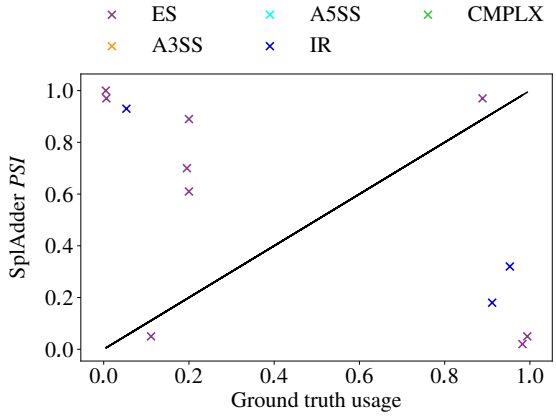
Figure 2.24: McSplicer results on spike-in RNA variants (SIRV) on 5 different SIRV samples. Ground truth splice site usages computed from known mixing ratios of SIRV isoforms are compared to usages estimated by McSplicer. Out of 38 variable splice sites, 26 belong to simple events and 12 belong to complex events. All these variable splice sites were correctly identified by StringTie and hence their usage estimated by McSplicer. Across the five samples, StringTie reported between 1 and 6 false splice sites within SIRV genes, which corresponds to a precision in splice site detection of approximately 99%. ES: exon skipping; A5SS: alternative 5' splice site; A3SS: alternative 3' splice site; CMPLX: complex event; IR: intron retention.



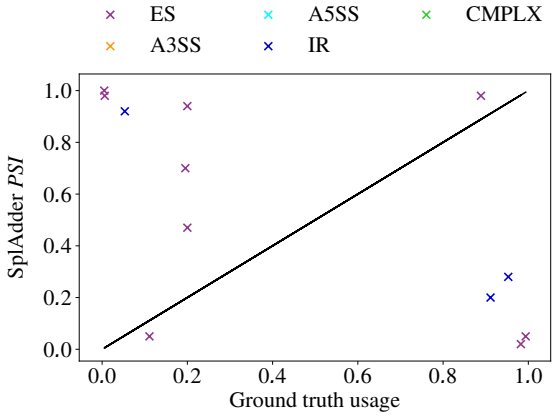
(a) Donor 1.



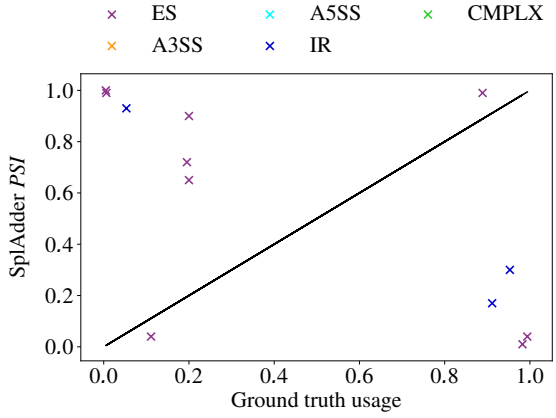
(b) Donor 2.



(c) Donor 3.

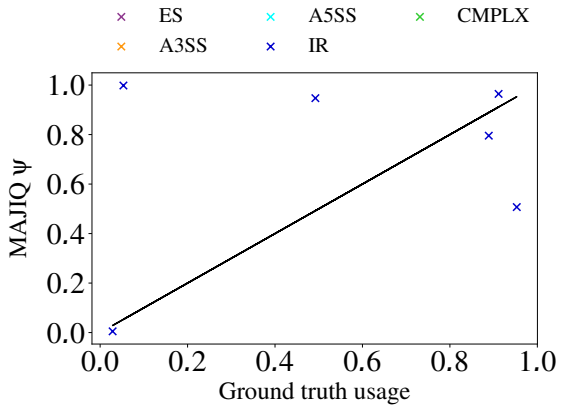


(d) Donor 4.

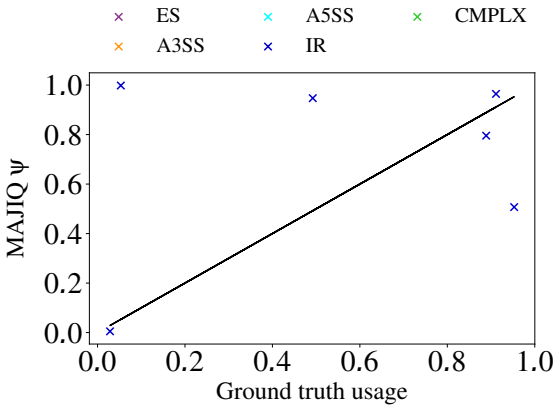


(e) Donor 5.

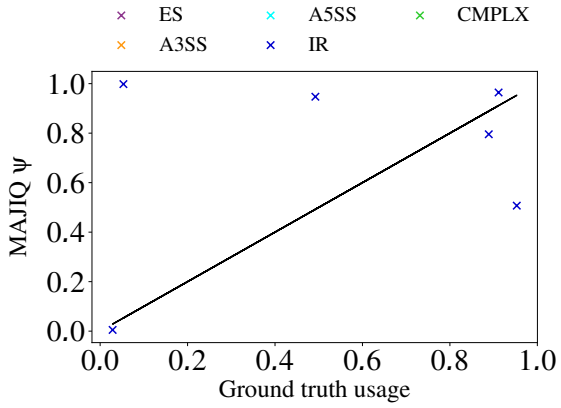
Figure 2.25: SplAdder results on spike-in RNA variants (SIRV) on 5 different SIRV samples. Ground truth splice site usages computed from known mixing ratios of SIRV isoforms are compared to usages estimated by SplAdder. Out of 38 variable splice sites, 26 belong to simple events and 12 belong to complex events. ES: exon skipping; A5SS: alternative 5' splice site; A3SS: alternative 3' splice site; CMPLX: complex event; IR: intron retention.



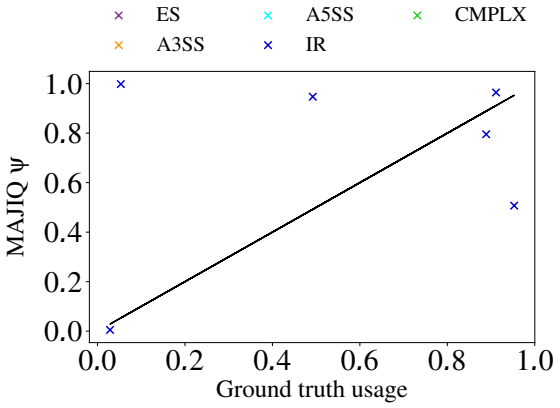
(a) Donor 1.



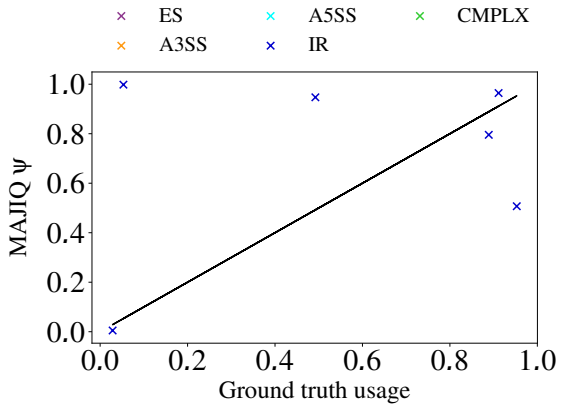
(b) Donor 2.



(c) Donor 3.



(d) Donor 4.



(e) Donor 5.

Figure 2.26: MAJIQ results on spike-in RNA variants (SIRV) on 5 different SIRV samples. Ground truth splice site usages computed from known mixing ratios of SIRV isoforms are compared to usages estimated by MAJIQ. Out of 38 variable splice sites, 26 belong to simple events and 12 belong to complex events. ES: exon skipping; A5SS: alternative 5' splice site; A3SS: alternative 3' splice site; CMPLX: complex event; IR: intron retention.

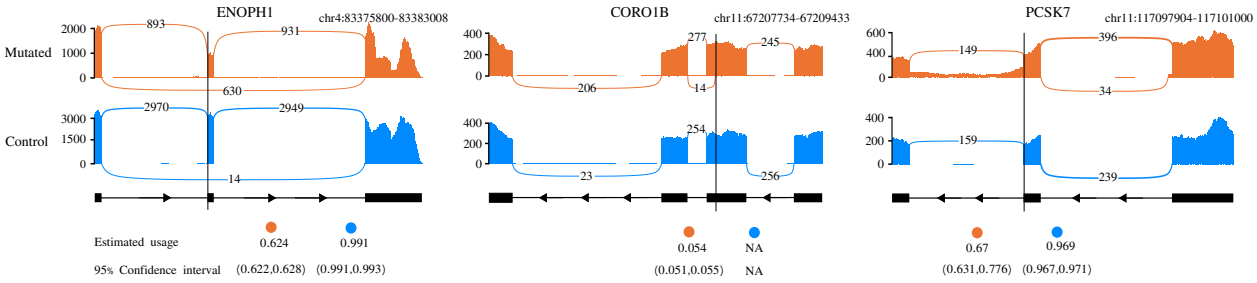


Figure 2.27: McSplicer splice site usage estimates and 95% bootstrapping confidence intervals for three disrupted splicing events reported in ASD patients versus control individuals. Variant locations are indicated by black vertical lines. Each plot illustrates the gene structure around the event with the precise genomic window specified on top, the read coverage and the junction read count. The Sashimi plots shown here are created using the ggsashimi tool (Garrido-Martín et al., 2018).

CHAPTER 3

RNA Ptr-Nets: Deep learning approach for predicting RNA splicing branchpoints

In the previous chapter we proposed a probabilistic model for estimating splice site usages. In this chapter we propose a deep learning approach for predicting RNA splicing branchpoints.

RNA splicing requires three main signals i.e., the donor splice site (i.e., 5'SS), the acceptor splice site (i.e., 3'SS), and the branchpoint (BP) site. These three signals work together in a two-step mechanism of RNA splicing. First, the pre-mRNA is cut at the donor site where the 5' end of the intron is joined to the BP site and generates a lariat intermediate molecule. Then, the last nucleotide of the intron at the 3' end is cut, and the two exons are joined together while the intron lariat is degenerated quickly, see Fig. 3.1A.

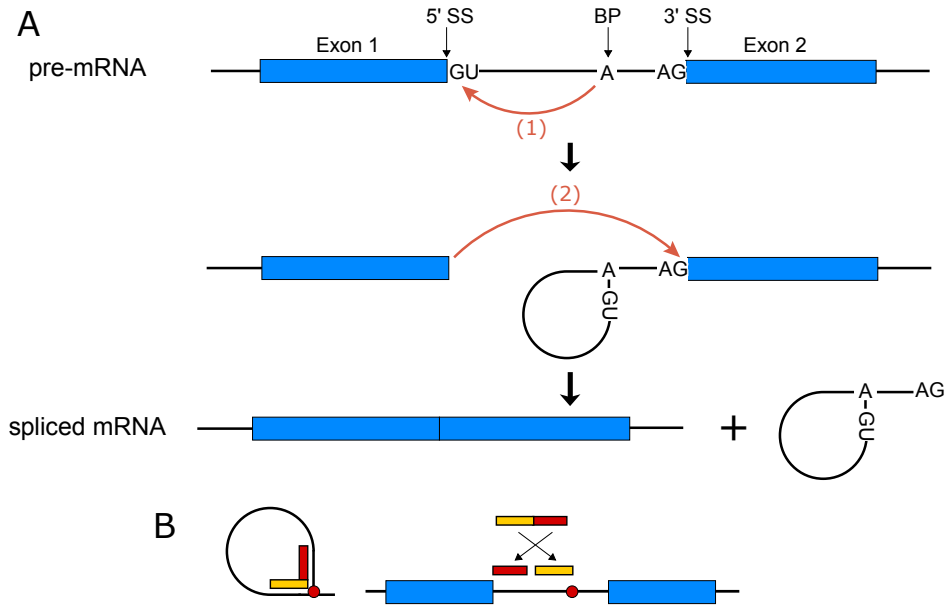


Figure 3.1: (A) The two-step mechanism of RNA splicing. Splicing begins with the dinucleotide sequence GU at the 5'SS and ends with the dinucleotide sequence AG at 3'SS. Branchpoints most frequently are adenosine. (B) RNA-seq reads spanning the 5'SS and BP junction can help determine the position of BPs.

Noticeably, BPs are mandatory signals in RNA splicing, equally important to 3'SSs and 5'SSs. Moreover, previous research shows that alternative BP selection plays a role in alternative splicing and diseases (Corvelo et al., 2010; Alsafadi et al., 2016). Splice donor and acceptor sites are well-studied in literature (Burset et al., 2001; Castelo and Guigó, 2004) and there exist several reliable tools for splice site prediction, e.g., Splicing Prediction in Consensus Elements (SPiCE) (Leman et al., 2018) which achieves 99.5 and 95.2% in sensitivity and specificity, respectively. However, the study of BP position has lagged behind due to two main challenges:

1. Experimental-wise: identifying BP positions is difficult in wet-lab experiments, because lariats exist for a short period of time as low-abundance RNAs.

2. Computational-wise: predicting BP positions is difficult and we lack an agreed upon genome-wide BP annotation due to two reasons:

    - Nucleotide motifs vary in the BP area (Pineda and Bradley, 2018).
    - Multiple BPs exist within the same intron. Previous research reported 95% of introns have multiple BPs with an estimate of 5 to 6 BPs per intron (Pineda and Bradley, 2018).

In theory, the position of BP can be determined using RNA-seq reads spanning the 5'SS and BP junctions, see Fig. 3.1B. However, the intron lariat is degenerated quickly during RNA splicing which makes such reads less frequent (Taggart et al., 2017). To date, human BP annotations are still incomplete.

Given the nature of the problem definition where only a small fraction of BPs is experimentally verified, different machine learning predictive models are trained using existing catalogues of human BPs, then they are used to predict the position of missing BPs. For example, SVM-BP employs Support Vector Machine algorithm (Corvelo et al., 2010), Branch Point Predictor (BPP) adapts a mixture model and expectation maximization algorithm (Zhang et al., 2017), and Branch-Pointer (Signal et al., 2018) is based on Gradient Boosting Machine (GBM) models. However, all these approaches require genome annotation and a decent amount of feature engineering prior to the training step.

Recently, with the emergence of deep learning methods, LaBranchoR (Long short-term memory network Branchpoint Retriever) was introduced as the state-of-the-art method for predicting a single BP for each 3'SS (Paggi and Bejerano, 2018). LaBranchoR uses two layers of bidirectional long short-term memory network (LSTM). LSTM shows superior performance when modeling sequential data such as RNA sequences (Lipton et al., 2015; Aggarwal et al., 2018). In their comparative benchmark analysis (Paggi and Bejerano, 2018), LaBranchoR showed superior performance compared to previous traditional machine learning methods. Moreover, LaBranchoR was applied in a recent study to examine cancer-related mutations of the splicing factor SF3B1 (Gupta et al., 2019).

Here, we introduce RNA Pointer Networks (RNA Ptr-Nets) model which is based on the novel deep learning architecture, Pointer Networks (Ptr-Nets) (Vinyals et al., 2015). The novelty of Ptr-Nets architecture is that it can handle output dictionaries whose size depends on the input sequence, such novelty allows Ptr-Nets model to be applied to the class of problems where outputs are discrete and correspond to positions in the input, e.g., sorting variable sized lists, and some combinatorial optimization problems such as convex hull, delaunay triangulation, travelling salesman problem (Vinyals et al., 2015). RNA Ptr-Nets model extends LaBranchoR to predict all the BPs associated with each 3'SS. Previous research showed that alternative BP selection plays a role in alternative

splicing and diseases, e.g., (Alsafadi et al., 2016) examined how alternative BPs affected splicing dysregulation in cancer. RNA Ptr-Nets model outputs pointers to BP positions with respect to each input sequence. The model does not require genome annotation or any feature engineering.

3.1 Materials and methods

3.1.1 Method

Given a training pair, (P, C^P), where $P = \{P_1, ..., P_n\}$ is a sequence of n vectors. P represents in our model a sequence of n nucleotides upstream of 3'SS. Each nucleotide is encoded by "one-hot encoding" scheme, where the nucleotide A is represented by the vector $(1,0,0,0)$, C is represented by the vector $(0,1,0,0)$, G is represented by the vector $(0,0,1,0)$, and T is represented by the vector $(0,0,0,1)$. The class labels, i.e., BP positions are indicated by $C^P = \{C_1, .., C_{m(P)}\}$, a sequence of $m(P)$ indices, i.e, positions of BPs with respect to the input sequence P, each index label is an integer number between 1 and n. Ptr-Nets model implements $p(C_i|C_1, ..., C_{i-1}, P)$ as follows:

$$u^i_j = v^T tanh(W_1 e_j + W_2 d_i) \quad j \in (1, .., n), i \in (1, .., m(P)) \tag{3.1}$$

$$p(C_i|C_1, ..., C_{i-1}, P) = softmax(u^i) \tag{3.2}$$

Where $(e_1, ..., e_n)$ and $(d_1, ..., d_{m(P)})$ are the encoder and decoder hidden states, respectively. v, W_1, and W_2 are the model's learnable parameters. And u^i is the attention mask over the input, the softmax function normalizes the attention vector u^i (of length n) to be an output probability distribution over the input indices. Then, we get the BP positions (i.e., output labels) form softmax output in equation 3.2, i.e., the index with the maximum probability represents a BP position (Vinyals et al., 2015).

RNA Ptr-Nets model adapts bidirectional LSTM networks for its encoder instead of the unidirectional LSTM proposed in the original model, Ptr-Nets. Bidirectional LSTM represents two LSTMs, i.e., forward and backward. The forward LSTM is fed the input sequence and the backward LSTM is fed the reverse of the input sequence. Forward state outputs are not connected to backward state inputs and vice versa, i.e., the input sequence and its reverse are processed independently. Then, the output of these two LSTMs are merged together, e.g., through concatenation, summation, averaging or multiplication (Schuster and Paliwal, 1997). In our model, RNA Ptr-Nets, we added forward and backward hidden states using learnable weights, then we applied element-wise hyperbolic tangent function $tanh(.)$. The output of $tanh(.)$ takes the place of the encoder hidden states $(e_1, ..., e_n)$ which we referred to in equation 3.1.

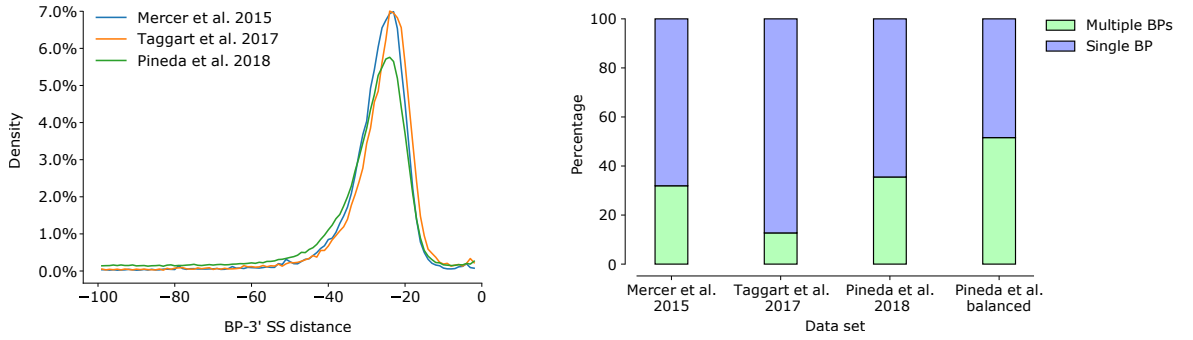
Furthermore, RNA Ptr-Nets model can be considered as an improvement over LabranchoR model. RNA Ptr-Nets model is based on an encoder-decoder architecture, where the encoder implements bidirectional LSTM and decoder implements unidirectional LSTM. On the other hand, LaBranchoR model uses two layers of bidirectional LSTM stacked on top of each other.

3.1.2 BP data sets

There are three publicly available BP data sets where large amount of RNA-seq data is leveraged to map splicing BPs in the human genome (Mercer et al., 2015), (Taggart et al., 2017), and (Pineda and Bradley, 2018), see Table 3.1. BPs are typically positioned between 18-45 nt upstream of the 3'SS, and this is consistent between different data sets as Fig.3.2(a) depicts.

Dataset	RNA-seq reads	BPs	Intron percentage
Mercer et al. 2015	$\sim 3 \times 10^9$	59,359	17.4%
Taggart et al. 2017	$\sim 11.3 \times 10^9$	36,078	16.8%
Pineda et al. 2018	$\sim 1.31 \times 10^{12}$	130,294	37%

Table 3.1: Existing human BP annotations. For each data set, we show the approximate number of RNA-seq reads used to provide the corresponding annotation, the total number BPs from high-confidence RNA-seq reads spanning the 5'SS-BP junction, and the percentage of introns with at least a single experimentally identified BP.



(a) The distance between BP and 3'SS based on existing annotations.

(b) The ratios of RNA sequences with single versus multiple BPs.

Figure 3.2: BP data sets characteristics.

Similar to the methodology followed by (Paggi and Bejerano, 2018), we extracted introns from the Gencode v19 annotations for all protein-coding genes. Then, we used Bedtools (Quinlan and Hall, 2010) to assign BPs to 3 'SSs, i.e., if a BP is located between 5-60 nucleotides (nt) upstream of a 3'SS, we assign the BP to that 3'SS. We extracted 70nt-long RNA sequences upstream of each 3'SS and we used them as the only input to RNA Ptr-Nets.

We leveraged Pineda et al. 2018 for training while reserving Mercer et al. 2015 and Taggart et al. 2017 for testing. Pineda et al. 2018 offers the most recent human BP catalogue with the largest number of high confidence BPs. However, during the initial experiments, we noticed that our model learned to predict almost always a single BP per sequence since the data set is unbalanced, i.e., more than 65% of RNA sequences are associated with a single BP. To address this issue, we generated Pineda balanced data set from the Pineda high confidence BPs set. In the balanced set, we included all sequences with multiple BPs and randomly selected 50% of sequences with a single BP. That way we guarantee that the model would observe almost equal number of RNA sequences with multiple BPs, see Fig. 3.2(b). We produced a training and validation split on Pineda balanced data set. The training-validation split was done by random and we followed $80/20$ rule, where 80% of the data was preserved for training and 20% of the data was preserved for validation. We tested the model on unseen RNA sequences from Mercer et al. 2015 and Taggart et al. 2017, see Fig. 3.3.

We measure the performance of our model using precision and recall metrics (equations 3.3 and 3.4).

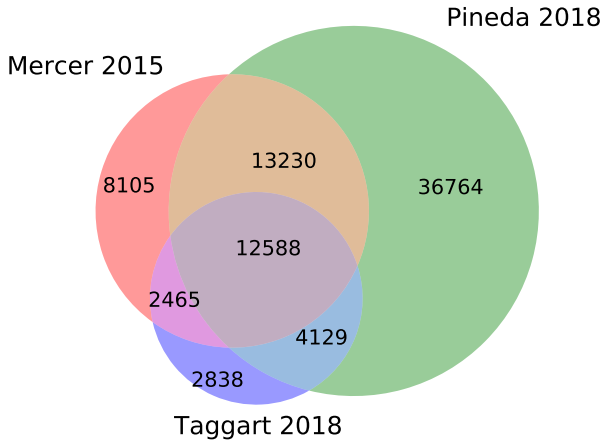


Figure 3.3: The total number of 70nt-long RNA sequences associated with all BPs reported in each data set. We evaluated RNA Ptr-Nets on mutually exclusive RNA sequences from Mercer et al. 2015 (8105 sequences) and Taggart et al. 2017 (2838 sequences).

$$Precision = \frac{TP}{TP+FP} \tag{3.3}$$

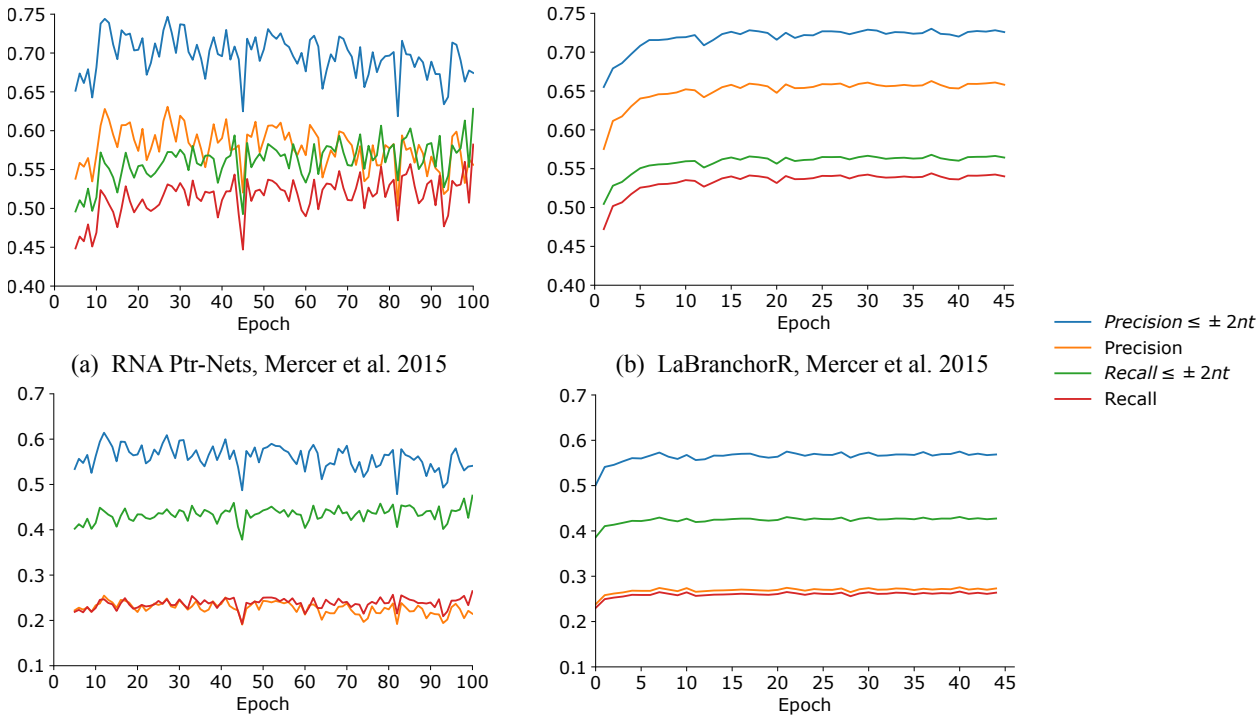
$$Recall = \frac{TP}{TP+FN} \tag{3.4}$$

Where TP refer to true positives , i.e., when the predicted BP positions match the experimental BPs. FP refers to false positives, i.e., when non-experimentally verified BP nucleotides are labeled as BPs by the model. FN refers to false negatives, the set of experimental BPs missed by the model.

3.2 Results

In this section, we present the results of RNA-Ptr Nets and LaBranchoR. Both methods are trained, validated and tested using the same data. LaBranchoR focuses on predicting the most likely BP associated with each 3'SS, while RNA Ptr-Nets model aims to predict all BPs associated with each 3'SS. Our task is more challenging, therefore, this comparative benchmark analysis aims solely to set a baseline and show how our model compares to the state-of-the-art method of LaBranchoR.

Figures 3.4(a) and (c) show the precision and recall of RNA Ptr-Nets on the test sets of Mercer et al. 2015 and Taggart et al. 2017. The performance of our model fluctuates and there is a noticeable trade-off between precision and recall, e.g., on Mercer et al. 2015 test set, earlier at epoch 27, the precision and recall values are $63.1\%, 53.2\%$, respectively. Later, at epoch 100, the precision and recall values are $55.6\%, 58.2\%$, respectively. The model in later epochs learns to predict multiple BPs per sequence at the cost of producing more false positives. On the other hand, LaBranchoR's performance is steady and increases slowly over time, see Fig. 3.4(b) and (d). LaBranchoR takes 45 epochs to converge during the training phase (i.e., on Pineda balanced set) and we notice beyond 45 epochs the performance on the validation set does not improve anymore while the performance on training set keeps improving. Using the model from epoch 45, the precision and recall are 65.8%, 54.0%, respectively on Mercer et al. 2015 test set. We notice that the predicted BPs often disagree with the experimental ones by small shifts, hence we additionally



(a) RNA Ptr-Nets, Mercer et al. 2015

(b) LaBranchorR, Mercer et al. 2015

(c) RNA Ptr-Nets, Taggart et al. 2017

(d) RNA Ptr-Nets, Taggart et al. 2017

Figure 3.4: Performance of RNA Ptr-Nets and LaBranchoR on the test sets of Mercer et al. 2015 and Taggart et al. 2017. In the 'precision $\leq \pm$2nt' and 'recall $\leq \pm$2nt' values, we consider a predicted BP as overlapping with an experimental BP if the predicted BP is within 2*nt* upstream or downstream of the experimental BPs.

report approximate values named 'precision $\leq \pm$2nt' and 'recall $\leq \pm$2nt', where we consider a predicted BP as overlapping with an experimental BP if the predicted BP lies within 2*nt* upstream or downstream of the experimental BPs. Table 3.2 summarizes the maximum precision and recall values achieved by each model and at which epochs.

	RNA Ptr-Nets	LaBranchoR
	Mercer et al. 2015	
Precision	0.631 (epoch 27)	**0.663** (epoch 36)
Precision $\leq \pm$2nt	**0.747** (epoch 27)	0.730 (epoch 36)
Recall	**0.582** (epoch 100)	0.544 (epoch 36)
Recall $\leq \pm$2nt	**0.628** (epoch 100)	0.568 (epoch 36)
	Taggart et al. 2017	
Precision	0.255 (epoch 12)	**0.276** (epoch 40)
Precision $\leq \pm$2nt	**0.614** (epoch 12)	0.575 (epoch 21)
Recall	0.264 (epoch 100)	**0.266** (epoch 40)
Recall $\leq \pm$2nt	**0.475** (epoch 100)	0.431 (epoch 40)

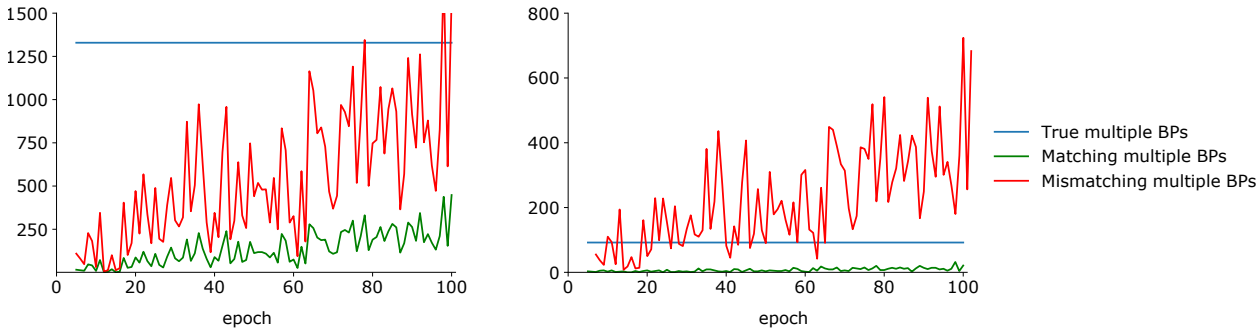
Table 3.2: Maximum precision and recall achieved by RNA Ptr-Nets and LaBranchoR together with epoch numbers on the test sets of Mercer et al. 2015 and Taggart et al. 2017. In 'precision $\leq \pm$2nt' and 'recall $\leq \pm$2nt' values, we consider a predicted BP as overlapping with an experimental BP if the predicted BP lies within 2*nt* upstream or downstream of the experimental BPs.

The Taggart BP sites generally disagreed with the predictions which resulted in lowering the performance of both models compared to Mercer BP sites, possibly due to overcompensation for nucleotide skipping in the denoising protocol adapted by (Taggart et al., 2017). Similar behavior is reported in (Paggi and Bejerano, 2018).

When considering only sequences with a single BP from Mercer et al. 2015 test set and using the final trained models of LaBranchoR and RNA Ptr-Nets, LaBranchor achieves 63.2% for both precision and recall, while RNA Ptr-Nets model achieves 52.7% and 66.2% for precision and recall, respectively. And when considering only sequences with a single BP from Taggart et al. 2017 test set, LaBranchor achieves 27.3% for both precision and recall while RNA Ptr-Nets model achieves 22.6% and 28.5% for precision and recall, respectively. We did additional experiments by adjusting the decoder of RNA Ptr-Nets to predict a single BP label instead of multiple labels. Similarly, we trained the model for 100 epochs. Using the final trained model we evaluated the performance on sequences with a single BP from Mercer et al. 2015 and Taggart et al. 2017 test sets. RNA Ptr-Nets model (with adjusted decoder) achieves 63.7% and 52.3% for precision and recall, respectively on Mercer et al. 2015 test sets. And it achieves 28.0% and 24.3% for precision and recall, respectively on Taggart et al. 2017 test sets.

3.2.1 RNA Ptr-Nets performance on sequences with multiple BPs

To further assess our model performance for predicting multiple BPs, we examine how often RNA Ptr-Nets model predicts multiple BP labels when indeed the ground truth data has multiple annotated BPs versus how often it predicts multiple BPs when there is only a single BP label. We anticipate that our model predicts more BPs than experimentally verified ones due to missing BP labels from exiting annotation, e.g., in Mercer data set only 17.4% of introns have one or more experimentally verified BP. Our model generates many false positives and we expect that a subset of these false positives to be missing labels from the experimentally verified BP data sets (see Fig. 3.5)



(a) Mercer et al. 2015

(b) Taggart et al. 2017

Figure 3.5: RNA Ptr-Nets model suffers from many false positives and negatives when dealing with sequences with multiple annotated BPs. Green line shows how often RNA Ptr-Nets model predicts multiple BPs when there are truly multiple annotated BPs, i.e., matching multiple BPs. Red line shows how often RNA Ptr-Nets model predicts multiple BPs when there is only a single annotated BP, i.e., mismatching multiple BPs. Blue line shows the ground truth, i.e., the total number of sequences with multiple BPs.

3.2.2 Further technical details and training time

RNA Ptr-Nets model was trained using the Adam optimizer, with a learning rate of 10^{-3}, using higher learning rate resulted in an unstable performance while lowering its value resulted in the model not being able to learn how to predict BP positions. We set the weight decay $\lambda = 0.5$, decreasing this value did not affect the performance of the model. We set the number of hidden nodes for both decoder and encoder to 256. We fixed the batch size to 256. The training is performed for 100 epochs. RNA Ptr-Nets model is implemented using PyTorch library (Paszke et al., 2019). LaBranchoR is implemented in Keras (Chollet, 2017) and its current implementation does not support batch processing, thus RNA Ptr-Nets model is much faster even when trained for more than twice the number of epochs. Using the same GPU machine, i.e., GeForce RTX 2080 with CUDA version 10.2 , LaBranchoR takes 6 days for training 45 epochs while RNA Ptr-Nets model takes ~ 5 hours for training 100 epochs on the same training set.

3.3 Discussion and conclusion

RNA Ptr-Nets implemented Ptr-Nets model which can handle output dictionaries whose size depends on the input sequence, the outputs of Ptr-Nets model can vary in length and they indicate pointers to the input indices. In this work, we aimed to develop a model for multiple BPs prediction. We generated a balanced training data set from Pineda high confidence BP set. The balanced training set helped the model observe almost equal number of RNA sequences with multiple as well as single BP labels.

RNA Ptr-Nets applies a softmax function to output probability distributions over input indices. LaBranchoR applies a sigmoid function to output a single prediction between zero and one for each nucleotide in the input sequence. RNA Ptr-Nets performance fluctuates and there is a noticeable trade-off between precision and recall. When compared to LaBranchoR, RNA Ptr-Nets often provided slightly higher recall at the cost of losing precision (i.e., it produced many false positives). Accurately predicting multiple BPs per 3'SS is more challenging than predicting a single BP site. RNA Ptr-Nets results showed that our model predicted more multiple BPs per sequence compared to the experimentally verified ones perhaps due to missing BP labels from exiting annotation, e.g., in Mercer data set only 17.4% of introns have one or more experimentally verified BP. Thus, we expect that a subset of these false positives to be missing labels from the original data sets.

CHAPTER 4

Conclusion and outlook

4.1 Conclusion

In Chapter 2, we introduced McSplicer, a novel probabilistic splicing model that estimates the usage of splice sites across expressed transcripts. McSplicer is not restricted to a set of predefined alternative splicing events. Our model is able to describe arbitrarily complex types of splicing patterns based on few, easy to interpret, parameters. We estimate these parameters, i.e. splice site usages, using all read data at once and demonstrate in simulation experiments that this yields more accurate estimates compared to other methods that use only reads directly supporting their parameters.

McSplicer model assumes the Markovian property across indicators (Z_i) for whether a segment is transcribed. This assumption allows for an efficient algorithm to estimate parameters of the model, but it potentially limits the ability of our method to model longer range dependencies such as between the recognition of 5' and 3' splice sites or between the removal of introns within transcripts. If true dependencies are longer than our model can describe, the individual estimators for splice site usages may still be accurate, but we expect transcript frequencies implied by our model to be less accurate (LeGault and Dewey, 2013). One way to model longer range dependencies is to use higher order Markov chains as long as the data provide sufficient information to estimate these dependencies.

The splice site usages computed by our method can be leveraged in various types of downstream analyses, e.g., we used McSplicer to quantify the effect size of splicing mutations in ASD patients where splice site usage as computed by McSplicer can be considered analogous to the "strength" of a splice site predicted by methods such as SplicePort (Dogan et al., 2007) from sequence-based features. Point mutations in the consensus splice site sequence can affect the strength of a splice site and result in the skipping of the exon or the activation of cryptic splice sites.

In Chapter 3, we introduced RNA Ptr-Nets, a deep learning method based on the novel neural network architecture, Pointer Networks (Ptr-Nets) (Vinyals et al., 2015). RNA Ptr-Nets extended LaBranchoR (Paggi and Bejerano, 2018) to predict all the BPs associated with each 3'SS. RNA Ptr-Nets took as input intronic sequences and outputed varied-length labels, i.e., pointers to BP positions with respect to each input sequence, further it did not require genome annotation or any

feature engineering. We extended BP prediction analysis to cover the most recent and largest dataset (Pineda and Bradley, 2018) which was not covered before. RNA Ptr-Nets performance fluctuated and there was a noticeable trade-off between precision and recall. When compared to LaBranchoR, RNA Ptr-Nets often provided slightly higher recall at the cost of losing precision (i.e., it produced many false positives). Accurately predicting multiple BPs per 3'SS is more challenging than predicting a single BP site.

RNA Ptr-Nets implemented Ptr-Nets model which can handle output dictionaries whose size depends on the input sequence, the outputs of Ptr-Nets model can vary in length and they indicate pointers to the input indices. In this work, we aimed to develop a model for multiple BPs prediction. We generated a balanced training data set from Pineda high confidence BP set. The balanced training set helped the model observe almost equal number of RNA sequences with multiple as well as single BP labels. RNA Ptr-Nets results showed that our model predicted multiple BPs per sequence compared to the experimentally verified ones perhaps due to missing BP labels from exiting annotation, e.g., in Mercer data set only 17.4% of introns have one or more experimentally verified BP. Thus, we expect that a subset of the false positives generated by our model to be missing labels from the original data sets

4.2 Outlooks

Outlook for McSplicer

We intend to extend McSplicer to include higher order Markov chains, then we can give McSplicer users the option to model longer range of dependencies between transcript segments, i.e., exons. This might also help in better estimating the probabilities of novel full-length transcripts. We also aim to leverage McSplicer in large-scale analysis on real-data from drosophila melanogaster at different time points. The goal is to examine muscle development phenotypes based on changes to alternative splicing especially intron retention defects.

On a separate note, we will extend McSplicer software to support multi-threaded programming, this way we provide more efficient and faster implementation to facilitate large-scale studies of alternative splicing.

Improving RNA Ptr-Nets

First, RNA Ptr-Nets model did not converge (i.e., the performance on the training as well as the validation data sets was fluctuating for 100 epochs). Therefore, further parameter tuning is required, e.g., using learning rate scheduler, where the value of learning rate decreases with time. If the model proves to be valid using existing annotation, we need to verify the validity of our model on existing diseased genes affected by competing BPs. Second, one way to improve the model's performance is to borrow ideas from existing language models such as the recent BERT model (Devlin et al., 2018) and to train such models to predict multiple BPs. Finally, we aim to apply our modified version of Ptr-Nets to other bioinformatics problems where the input sequences vary in length and outputs are pointers to input indices.

APPENDIX A

McSplicer Supplementary Material

A.1 Benchmarks

Tools and parameters

Polyester simulator

We used simulated data to evaluate McSplicer accuracy. As mentioned in the main text, we used Polyester simulator (version 1.16.0) to simulate RNA-seq reads from human transcripts (Ensembl release 91). For the three different sequencing depths, we used the software with its default parameters, and we ran it under the following environment:

```
R version 3.5.2 (2018-12-20)
Platform: x86_64-redhat-linux-gnu (64-bit)
Running under: Scientific Linux 7.5 (Nitrogen)
```

As previously mentioned, we provided Polyester with ground truth abundances computed by running RSEM quantification tool Li and Dewey (2011) on RNA-seq data from SRA data set SRR6987574 [1]. Then, we randomly selected a set of 1000 genes which have at least two expressed transcripts and have sufficiently high expression, i.e., gene-level read count per kilobase > 500.

STAR aligner

The simulated reads were mapped to the human reference genome (GRCh38.91) by running STAR (version 2.5.4b) Dobin et al. (2013) with the following parameters:

```
--runMode alignReads
--outSAMtype BAM SortedByCoordinate
--sjdbGTFfile Homo_sapiens.GRCh38.91.gtf
--runThreadN 16
--readFilesIn {ployester_output.fasta}
--outFileNamePrefix {output_prefix}
--genomeDir {genome_directory}
--outSAMstrandField intronMotif
--sjdbGTFfile
```

The remaining set of parameters were left to the default values.
For indexing the resulting BAM files we used Samtools (version 0.1.8) Li et al. (2009).

StringTie

We ran StringTie Pertea et al. (2015) (version 1.3.4d) with genome-guided mode enabled (-G option) and provided Ensembl annotation release 91. The remaining parameters of StringTie were left to the default values.

SplAdder

We ran SplAdder (version 1.2.0) with the following set of parameters for benchmarking on simulated data:

[1] http://www.ncbi.nlm.nih.gov/sra

```
--bams {bam_files}
--annotation {annotation_gtf}
--merge_strat merge_graphs
--event_types exon_skip,intron_retention,alt_3prime,
alt_5prime,mult_exon_skip
--confidence 2
--pyproc n
--compress_text n
--ignore_mismatches y
--outdir {output_directory}
```

We set the *confidence* parameter to 1 when running SplAdder on the SIRV dataset in order to detect novel events.

MAJIQ

We ran MAJIQ (version 2.0) with default parameters but with de novo option disabled, i.e., $disable-denovo$ for all benchmarks on simulated data. We noticed many false positive events when running MAJIQ without the $disable-denovo$ argument (i.e., enabling de novo mode). We enable de novo mode again when evaluating MAJIQ on SIRV data sets to detect as many novel events as possible.

PSGInfer

To compute edge weight estimates using PSGInfer, we followed two steps. First, we executed the command psg_prepare_reference to generate a reference splice graph from annotated transcripts (Ensembl annotation release 91), and we configured it to generate a line graph since it is computationally more efficient than other types of graphs yet provides accurate estimates of edge weights LeGault and Dewey (2013). Second, we ran psg_infer_frequencies to map RNA-seq reads to the splice graphs generated in the first step and to estimate the weights of graph edges. PSGInfer uses Bowtie Langmead et al. (2009) internally for RNA-seq read mapping. We ran the latest version of PSGInfer 1.2.1 and a compatible version of Bowtie 1.3.0.

```
--annotations {annotation_gtf}
--genome-dir {chromosome_FASTA_files_dir}
-l 100 {max_read_length}
-k 0 {order_of_PSG}
--num-threads 72
```

Bibliography

Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

Gael P Alamancos, Amadís Pagès, Juan L Trincado, Nicolás Bellora, and Eduardo Eyras. Leveraging transcript quantification for fast computation of alternative splicing profiles. *Rna*, 21(9): 1521–1531, 2015.

Samar Alsafadi, Alexandre Houy, Aude Battistella, Tatiana Popova, Michel Wassef, Emilie Henry, Franck Tirode, Angelos Constantinou, Sophie Piperno-Neumann, Sergio Roman-Roman, et al. Cancer-associated sf3b1 mutations affect alternative splicing by promoting alternative branchpoint usage. *Nature communications*, 7(1):1–12, 2016.

Abramowicz Anna and Gos Monika. Splicing mutations in human genetic disorders: examples, detection, and confirmation. *Journal of applied genetics*, 59(3):253–268, 2018.

Ulrich Braunschweig, Nuno L Barbosa-Morais, Qun Pan, Emil N Nachman, Babak Alipanahi, Thomas Gonatopoulos-Pournatzis, Brendan Frey, Manuel Irimia, and Benjamin J Blencowe. Widespread intron retention in mammals functionally tunes transcriptomes. *Genome research*, 24(11):1774–1786, 2014.

Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525, 2016.

Angela N Brooks, Li Yang, Michael O Duff, Kasper D Hansen, Jung W Park, Sandrine Dudoit, Steven E Brenner, and Brenton R Graveley. Conservation of an rna regulatory map between drosophila and mammals. *Genome research*, 21(2):193–202, 2011.

M Burset, Igor A Seledtsov, and Victor V Solovyev. Splicedb: database of canonical and non-canonical mammalian splice sites. *Nucleic acids research*, 29(1):255–259, 2001.

Stefan Canzar, Sandro Andreotti, David Weese, Knut Reinert, and Gunnar W Klau. Cidane: comprehensive isoform discovery and abundance estimation. *Genome biology*, 17(1):16, 2016.

Robert Castelo and Roderic Guigó. Splice site identification by idl bns. *Bioinformatics*, 20 (suppl_1):i69–i76, 2004.

Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2017.

André Corvelo, Martina Hallegger, Christopher WJ Smith, and Eduardo Eyras. Genome-wide association between branch point properties and alternative splicing. *PLoS computational biology*, 6(11):e1001016, 2010.

Luca Denti, Raffaella Rizzi, Stefano Beretta, Gianluca Della Vedova, Marco Previtali, and Paola Bonizzoni. ASGAL: aligning RNA-Seq data to a splicing graph to detect novel alternative splicing events. *BMC Bioinformatics*, 19(1):444, Nov 2018. ISSN 1471-2105. doi: 10.1186/s12859-018-2436-3. URL https://doi.org/10.1186/s12859-018-2436-3.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

Rezarta Islamaj Dogan, Lise Getoor, W John Wilbur, and Stephen M Mount. Spliceport—an interactive splice-site analysis tool. *Nucleic acids research*, 35(suppl_2):W285–W291, 2007.

Sylvain Foissac and Michael Sammeth. Astalavista: dynamic and flexible analysis of alternative splicing events in custom gene datasets. *Nucleic acids research*, 35(suppl_2):W297–W299, 2007.

Alyssa C Frazee, Andrew E Jaffe, Ben Langmead, and Jeffrey T Leek. Polyester: simulating rna-seq datasets with differential transcript expression. *Bioinformatics*, 31(17):2778–2784, 2015.

Diego Garrido-Martín, Emilio Palumbo, Roderic Guigó, and Alessandra Breschi. ggsashimi: Sashimi plot revised for browser-and annotation-independent splicing visualization. *PLoS computational biology*, 14(8):e1006360, 2018.

Abhishek K Gupta, Tushar Murthy, Kiran V Paul, Oscar Ramirez, Joseph B Fisher, Sridhar Rao, Alexander B Rosenberg, Georg Seelig, Alex C Minella, and Manoj M Pillai. Degenerate minigene library analysis enables identification of altered branch point utilization by mutant splicing factor 3b1 (sf3b1). *Nucleic acids research*, 47(2):970–980, 2019.

Katharina E. Hayer, Gregory R. Grant, Angel Pizarro, John B. Hogenesch, and Nicholas F. Lahens. Benchmark analysis of algorithms for determining and quantifying full-length mRNA splice forms from RNA-seq data. *Bioinformatics*, 31(24):3938–3945, 09 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv488. URL https://doi.org/10.1093/bioinformatics/btv488.

Steffen Heber, Max Alekseyev, Sing-Hoi Sze, Haixu Tang, and Pavel A Pevzner. Splicing graphs and est assembly problem. *Bioinformatics*, 18(suppl_1):S181–S188, 2002.

Florian Hoss, James L Mueller, Francisca Rojas Ringeling, Juan F Rodriguez-Alcazar, Rebecca Brinkschulte, Gerald Seifert, Rainer Stahl, Lori Broderick, Chris D Putnam, Richard D Kolodner, et al. Alternative splicing regulates stochastic nlrp3 activity. *Nature communications*, 10(1):1–13, 2019.

Kishore Jaganathan, Sofia Kyriazopoulou Panagiotopoulou, Jeremy F McRae, Siavash Fazel Darbandi, David Knowles, Yang I Li, Jack A Kosmicki, Juan Arbelaez, Wenwu Cui, Grace B

Schwartz, et al. Predicting splicing from primary sequence with deep learning. *Cell*, 176(3): 535–548, 2019.

Andre Kahles, Cheng Soon Ong, Yi Zhong, and Gunnar Rätsch. Spladder: identification, quantification and testing of alternative splicing events from rna-seq data. *Bioinformatics*, 32(12): 1840–1847, 2016.

André Kahles, Kjong-Van Lehmann, Nora C Toussaint, Matthias Hüser, Stefan G Stark, Timo Sachsenberg, Oliver Stegle, Oliver Kohlbacher, Chris Sander, Samantha J Caesar-Johnson, et al. Comprehensive analysis of alternative splicing across tumors from 8,705 patients. *Cancer cell*, 34(2):211–224, 2018.

Yarden Katz, Eric T Wang, Edoardo M Airoldi, and Christopher B Burge. Analysis and design of rna sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009, 2010.

Daehwan Kim, Ben Langmead, and Steven L Salzberg. Hisat: a fast spliced aligner with low memory requirements. *Nature methods*, 12(4):357–360, 2015.

Vincent Lacroix, Michael Sammeth, Roderic Guigo, and Anne Bergeron. Exact transcriptome reconstruction from short sequence reads. In *International Workshop on Algorithms in Bioinformatics*, pages 50–63. Springer, 2008.

Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biology*, 10(3):R25, 2009.

Laura H LeGault and Colin N Dewey. Inference of alternative splicing from rna-seq data with probabilistic splice graphs. *Bioinformatics*, 29(18):2300–2310, 2013.

Raphael Leman, Pascaline Gaildrat, Gérald Le Gac, Chandran Ka, Yann Fichou, Marie-Pierre Audrezet, Virginie Caux-Moncoutier, Sandrine M Caputo, Nadia Boutry-Kryza, Melanie Leone, et al. Novel diagnostic tool for prediction of variant spliceogenicity derived from a set of 395 combined in silico/in vitro studies: an international collaborative effort. *Nucleic acids research*, 46(15):7913–7923, 2018.

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.

Bo Li, Victor Ruotti, Ron M Stewart, James A Thomson, and Colin N Dewey. Rna-seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493–500, 2010.

Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

Yang I Li, David A Knowles, Jack Humphrey, Alvaro N Barbeira, Scott P Dickinson, Hae Kyung Im, and Jonathan K Pritchard. Annotation-free quantification of rna splicing using leafcutter. *Nature genetics*, 50(1):151–158, 2018.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Arfa Mehmood, Asta Laiho, Mikko S Venäläinen, Aidan J McGlinchey, Ning Wang, and Laura L Elo. Systematic evaluation of differential splicing tools for rna-seq studies. *Briefings in Bioinformatics*, 2019.

Tim R Mercer, Michael B Clark, Stacey B Andersen, Marion E Brunck, Wilfried Haerty, Joanna Crawford, Ryan J Taft, Lars K Nielsen, Marcel E Dinger, and John S Mattick. Genome-wide discovery of human splicing branchpoints. *Genome research*, 25(2):290–303, 2015.

Vasilis Ntranos, Govinda M Kamath, Jesse M Zhang, Lior Pachter, and N Tse David. Fast and accurate single-cell rna-seq analysis by clustering of transcript-compatibility counts. *Genome biology*, 17(1):112, 2016.

Fernando Carrillo Oesterreich, Hugo Bowne-Anderson, and Jonathon Howard. The contribution of alternative splicing probability to the coding expansion of the genome. *bioRxiv*, 2016. doi: 10.1101/048124. URL `https://www.biorxiv.org/content/early/2016/04/11/048124`.

Joseph M Paggi and Gill Bejerano. A sequence-based, deep learning model accurately predicts rna splicing branchpoints. *RNA*, 24(12):1647–1658, 2018.

Qun Pan, Ofer Shai, Leo J. Lee, Brendan J. Frey, and Benjamin J. Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, 40:1413–1415, Nov 2008. URL `https://doi.org/10.1038/ng.259`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Ellis Patrick, Michael Buckley, and Yee Hwa Yang. Estimation of data-specific constitutive exons with rna-seq data. *BMC bioinformatics*, 14(1):1–10, 2013.

Lukas Paul, Petra Kubala, Gudrun Horner, Michael Ante, Igor Hollaender, Seitz Alexander, and Torsten Reda. Sirvs: Spike-in rna variants as external isoform controls in rna-sequencing. *bioRxiv*, page 080747, 2016.

Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. Stringtie enables improved reconstruction of a transcriptome from rna-seq reads. *Nature biotechnology*, 33(3):290, 2015.

Jose Mario Bello Pineda and Robert K Bradley. Most human introns are recognized via multiple and tissue-specific branchpoints. *Genes & development*, 32(7-8):577–591, 2018.

Aaron R Quinlan and Ira M Hall. Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.

AS Ramalho, S Beck, Deborah Penque, Tanja Gonska, HH Seydewitz, Marcus Mall, and MD Amaral. Transcript analysis of the cystic fibrosis splicing mutation 1525-1g¿ a shows use of multiple alternative splicing sites and suggests a putative role of exonic splicing enhancers. *Journal of medical genetics*, 40(7):e88–e88, 2003.

Mark F Rogers, Julie Thomas, Anireddy SN Reddy, and Asa Ben-Hur. Splicegrapher: detecting patterns of alternative splicing from rna-seq data in the context of gene models and est data. *Genome biology*, 13(1):R4, 2012.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Marvin B Shapiro and Periannan Senapathy. Rna splice junctions of different classes of eukaryotes: sequence statistics and functional implications in gene expression. *Nucleic acids research*, 15 (17):7155–7174, 1987.

Shihao Shen, Juw Won Park, Zhi-xiang Lu, Lan Lin, Michael D Henry, Ying Nian Wu, Qing Zhou, and Yi Xing. rmats: robust and flexible detection of differential alternative splicing from replicate rna-seq data. *Proceedings of the National Academy of Sciences*, 111(51):E5593–E5601, 2014.

Bethany Signal, Brian S Gloss, Marcel E Dinger, and Tim R Mercer. Machine learning annotation of human branchpoints. *Bioinformatics*, 34(6):920–927, 2018.

Alicia C Smart, Claire A Margolis, Harold Pimentel, Meng Xiao He, Diana Miao, Dennis Adeegbe, Tim Fugmann, Kwok-Kin Wong, and Eliezer M Van Allen. Intron retention is a source of neoepitopes in cancer. *Nature biotechnology*, 36(11):1056–1058, 2018.

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.

Li Song and Liliana Florea. Class: constrained transcript assembly of rna-seq reads. *BMC Bioinformatics*, 14(5):S14, Apr 2013. ISSN 1471-2105. doi: 10.1186/1471-2105-14-S5-S14. URL `https://doi.org/10.1186/1471-2105-14-S5-S14`.

Timothy Sterne-Weiler, Robert J Weatheritt, Andrew Best, Kevin CH Ha, and Benjamin J Blencowe. Whippet: an efficient method for the detection and quantification of alternative splicing reveals extensive transcriptomic complexity. *bioRxiv*, page 158519, 2017.

Allison J Taggart, Chien-Ling Lin, Barsha Shrestha, Claire Heintzelman, Seongwon Kim, and William G Fairbrother. Large-scale analysis of branchpoint usage across species and cell lines. *Genome research*, 27(4):639–649, 2017.

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.

Jorge Vaquero-Garcia, Alejandro Barrera, Matthew R Gazzara, Juan Gonzalez-Vallinas, Nicholas F Lahens, John B Hogenesch, Kristen W Lynch, and Yoseph Barash. A new view of transcriptome complexity and regulation through the lens of local splicing variations. *elife*, 5: e11752, 2016.

Jorge Vaquero-Garcia, Scott Norton, and Yoseph Barash. Leafcutter vs. majiq and comparing software in the fast-moving field of genomics. *bioRxiv*, page 463927, 2018.

Julian P Venables, Roscoe Klinck, Anne Bramard, Lyna Inkel, Genevieve Dufresne-Martin, ChuShin Koh, Julien Gervais-Bird, Elvy Lapointe, Ulrike Froehlich, Mathieu Durand, et al. Identification of alternative splicing markers for breast cancer. *Cancer research*, 68(22):9525–9531, 2008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *arXiv preprint arXiv:1506.03134*, 2015.

Qingqing Wang and Donald C Rio. Jum is a computational method for comprehensive annotation-free analysis of alternative pre-mrna splicing patterns. *Proceedings of the National Academy of Sciences*, 115(35):E8181–E8190, 2018.

Feng Xiong, Jianjun Gao, Jun Li, Yun Liu, Guoyin Feng, Wenli Fang, Hongfen Chang, Jiang Xie, Haitao Zheng, Tingyu Li, et al. Noncanonical and canonical splice sites: a novel mutation at the rare noncanonical splice-donor cut site (ivs4+ 1a¿ g) of sedl causes variable splicing isoforms in x-linked spondyloepiphyseal dysplasia tarda. *European journal of human genetics*, 17(4): 510–516, 2009.

Qing Zhang, Xiaodan Fan, Yejun Wang, Ming-an Sun, Jianlin Shao, and Dianjing Guo. Bpp: a sequence-based algorithm for branch point prediction. *Bioinformatics*, 33(20):3166–3172, 2017.