# *SplAdder:* Identification, quantification and testing of alternative splicing events from RNA-Seq data

André Kahles,[1] Cheng Soon Ong,[2] and Gunnar Rätsch[1]

[1]Computational Biology Center, Sloan-Kettering Institute, 1275 York Ave, New York, NY 10065, USA
[2]NICTA, Canberra Research Laboratory, Tower A, 7 London Circuit, Canberra ACT 2601, Australia

**Abstract**

### Motivation

Understanding the occurrence and regulation of alternative splicing (AS) is a key task towards explaining the regulatory processes that help to shape the complex transcriptomes of higher eukaryotes. With the advent of high-throughput sequencing of RNA, the diversity of AS transcripts could be measured at an unprecedented depth. Although the catalog of known AS events has ever grown since, novel isoforms are commonly observed when working with less annotated organisms, in the context of diseases, or within large populations. Whereas an identification of complete isoforms is technically challenging or expensive, focussing on single splicing events as a proxy for transcriptome characteristics is fruitful for differential analyses.

### Results

We present *SplAdder*, an alternative splicing toolbox, that takes RNA-Seq alignments and an annotation file as input to i) augment the annotation based on RNA-Seq evidence, ii) identify alternative splicing events present in the augmented annotation graph, iii) quantify and confirm these events based on the RNA-Seq data, and iv) test for significant quantitative differences.

### Availability

Source code and documentation are available for download at `github.com/ratschlab/spladder`. Example data, introductory information and a small tutorial are accessible at `bioweb.me/spladder`.

### Contact

`akahles@cbio.mskcc.org`, `raetsch@cbio.mskcc.org`

## 1 Introduction

Alternative splicing (AS) is an mRNA processing mechanism that cuts and re-joins maturing mRNA in a highly regulated manner, thereby increasing transcriptome complexity. Depending on the organism, up to 95% of expressed genes are transcribed in multiple isoforms [6, 8]. Although these isoforms might never coexist at the same time and place, each one of them can be essential for cell differentiation, development or within signaling processes. Thus, the two major challenges in computational transcriptome analysis are complexity and completeness. In *SplAdder*, we leverage evidence from RNA-Seq data to compute a more complete representation of the splicing diversity within a sample and tackle the complexity with a reduction to alternative splicing events instead of full transcripts. We provide implementations for *SplAdder* in Matlab and Python that contain all features described below and produce identical results. However, future development will

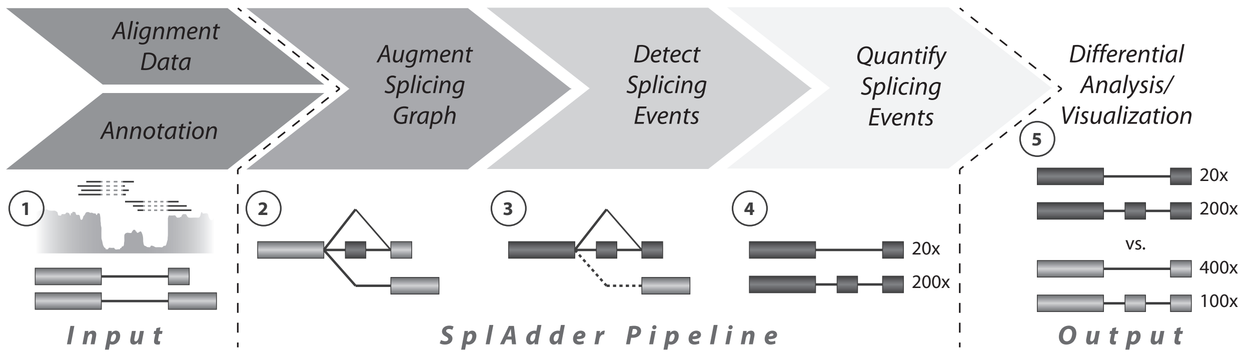

Figure 1: ***SplAdder* Analysis Flowchart** The main steps of the *SplAdder* workflow consist of the integration of annotation information and RNA-Seq data (1) into an augmented splicing graph (2), the extraction of splicing events from that graph (3), the quantification of the extracted events (4) and, optionally, the differential analysis between samples and visualization (5).

focus in the Python implementation for reasons of accessibility. All input formats follow the standardized formats for alignments and annotation such as BAM and GFF. For examples, uses cases and information regarding the user interface we refer to the supplementary website and the user documentation.

## 2 Approach

The *SplAdder* pipeline consists of multiple steps that convert any given annotation into a splicing graph, enrich that graph with splicing evidence from RNA-Seq samples, identify splicing events from the augmented graph and use the given RNA-Seq data to quantify the single events (Figure 1). Optionally, the quantifications can then be used for differential analysis.

### 2.1 Construct Augmented Splicing Graph

From a given annotation, all transcript isoforms of a gene are parsed into a single splicing graph representation, where exons are represented as nodes and introns as edges. Exons, identified a combination of start and end position, that are shared by several isoforms will be represented as a single node. By breaking up long range dependencies in the transcripts the splicing complexity is drastically reduced, speeding up any further processing. Once the annotation is parsed, we add additional nodes and edges to the graph, if we find sufficient evidence in the given RNA-Seq alignment data. In successive steps, different types of AS elements are added to the graph. For a summary of all augmentations and their respective support criteria we refer to Section A in the Supplementary Material. Furthermore, to reduce false positive edges in the graph, *SplAdder* allows for various strategies for removing edges based on evidence in multiple RNA-Seq samples.

### 2.2 Detect Alternative Splicing Events

Based on the augmented splicing graph, it is effortless to extract various classes of AS events as subsets of connected nodes. *SplAdder* currently supports the following event types: exon skip, intron retention, alternative 3' and alternative 5' splice sites as well as multiple exon skips. Each event is then represented as a "mini-gene" consisting of two isoforms minimally describing the alternatives of the event. Overlapping events that share the same intron coordinates and do only differ in the flanking exon ends are merged into a short common representation.

### 2.3 Quantify Splicing Events

The event set identified from the splicing graph is then quantified using the given read alignment data. For each node (exon) the mean coverage and for each edge (intron) the number of supporting spliced alignments is reported. How an alignment is used for quantification depends on user adjustable filter criteria that ensure that only alignments of sufficient confidence are counted. To speed up the counting process, the splicing graph is internally represented as graph of non-overlapping exonic segments that are then quantified. Thus, no exonic position needs to be quantified twice.

### 2.4 Differential Analysis and Visualization

If the set of input samples can be separated into two groups representing different conditions are sample types, differential testing can be applied to the counts generated from these samples. *SplAdder* does not implement a differential testing routine itself, but employs rDiff [2]. *SplAdder* also provides means for rich visualization of the RNA-seq read coverage of exonic positions and of intron junctions. Visualization allows for effective visual inspection of identified alternative splicing events in light of primary read data.

## 3 Evaluation and Applications

The *SplAdder* approach has been successfully used in various biological studies in *Arabidopsis thaliana* [1, 3] as well as in the context of large-scale cancer projects with several thousand RNA-seq libraries [9]. Here, we will provide an evaluation based on simulated data to give an accurate measure of performance. Briefly, we have used *FluxSimulator* [4] to simulate RNA-Seq reads from $5,000$ randomly selected human genes. Only the first annotated transcript was provided to *SplAdder* with the task to recover the omitted information based on the read data. *SplAdder* predicts both exons and introns with very high accuracy (cf. Figure 2). Details regarding data simulation and evaluation of results as well as different ways to visualize the splicing data can be found in Section C of the Supplemental Material.

## 4 Conclusion

We present *SplAdder*, a versatile tool for the large-scale analysis of alternative splicing events based on RNA-Seq data. It is straightforward to use and can be easily deployed in a high performance computing framework. *SplAdder* has been successfully applied to splicing analysis in various organisms, can be readily applied to datasets of thousands of samples and shows high accuracy in an evaluation on simulated data.

### Acknowledgements

The authors are grateful to Vipin T Sreedharan for providing code to convert annotation files and to Andreas Wachter for valuable discussions and feedback to the software. *Funding* was provided

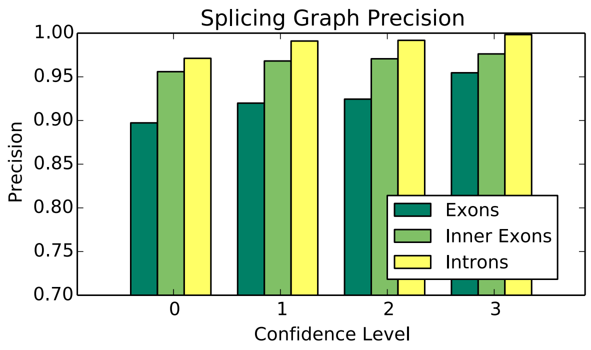

Figure 2: ***SplAdder* Evaluation Results** Result of the performance evaluation based on simulated read data. Precision levels for nodes and edges of the graph for the four *SplAdder* confidence levels.

by the Max Planck Society, Memorial Sloan Kettering Cancer Center and by the German Research Foundation (RA1894/2-1) and the Lucille Castori Center for Microbes, Inflammation, and Cancer (No. 223316).

## A Splicing Graph Augmentation

### A.1 Definitions and Notation

A given gene annotation can be represented as a set of linear directed graphs. Assume gene $g$ is given and has $k$ different isoforms $j_1, \ldots, j_k \in J_g$, where $J_g$ is the set of all isoforms of gene $g$. As we consider each gene $g$ individually, we will omit the index $g$ wherever possible in order to keep the notation uncluttered. Each isoform consists of a set of exons that are connected by introns. Each exon can be uniquely identified by its start and its end. We thus represent all exons as coordinate pairs of their start end stop position:

$$v = (\text{start}, \text{stop}) = (v_{\text{start}}, v_{\text{stop}}) \in \mathbb{N}^2.$$

Although further coordinate information like chromosome and strand are used in the program implementation, we will limit this description to an identification by start and stop for simplicity. The exons of each isoform $j_i$ can then be represented as a node set $V_i := \{v_{i,1}, \ldots, v_{i,m_i}\}$ with $1 \leq i \leq k$ and $m_i$ as the number of exons in isoform $j_i$. As transcripts have a direction (the exons within a transcripts follow a strict order), we require, that the index of the nodes reflects the order of the exons in the transcript. As no two exons in a transcript overlap by definition, this order is implied by $v_{\text{start}}$ and $v_{\text{stop}}$. We then define the edge set of isoform $j_i$ as

$$E_i := \bigcup_{1 \leq s < m_i} \{(v_{i,s}, v_{i,s+1}) \mid v_{i,s}, v_{i,s+1} \in V_i\} \subset V_i \times V_i$$

with $1 \leq i \leq k$. The pair $(V_i, E_i)$ forms the directed isoform graph of isoform $j_i$.

Next, we define the set of exons occurring in *any* isoform $j_i$ as $V$. As the single exons are uniquely identified by their coordinates, we can write $V := \bigcup_{i=1}^{k} V_i$. Hence, we define the set of all edges as

$$E := \bigcup_{i=1}^{k} E_i \subset V \times V.$$

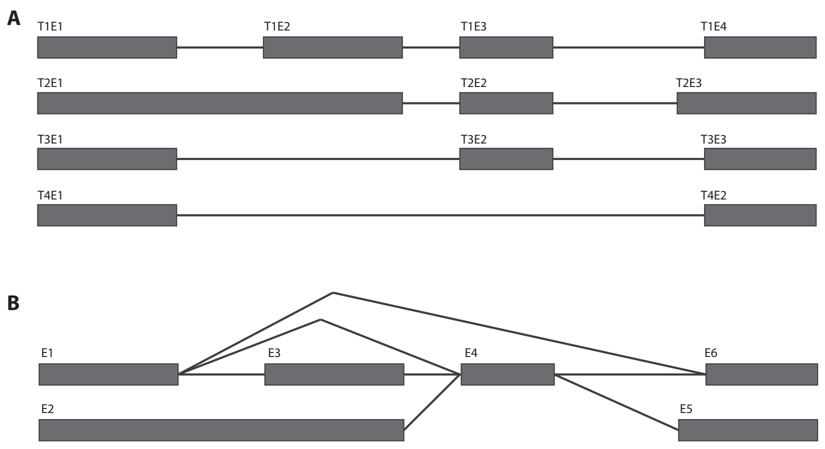

Figure 3: Example case for the construction of the splicing graph. ***A:*** Set of four different transcript isoforms. Exons are depicted as gray boxes and introns as solid lines. Labels T$i$E$j$ denote exon $j$ in transcript $i$. ***B:*** Splicing graph representation of the same four isoforms. Exons occurring in multiple isoforms are collapsed into a single exon in the graph.

Note that only already existing edges are merged, preserving any existing order of nodes. The pair $G = (V, E)$ is a directed acyclic graph and is called *splicing graph* representation of a gene. Figure 3 illustrates how a set of four isoforms is collapsed into a splicing graph.

We define the *in-degree* and the *out-degree* of a node as the number of its incoming and outgoing edges, respectively. We further define a node to be *start-terminal*, if its in-degree is zero and *end-terminal* if its out-degree is zero. Each isoform can now be represented as a path through the splicing graph, beginning at a start-terminal node and ending at an end-terminal node.

Although the splicing graph representation resolves many redundancies and thus can efficiently store large numbers of different but mostly overlapping isoforms, this comes at the cost of information loss. Long range dependencies between single exons are not preserved. An example of this is provided in Figure 3, Panel B. Although exon T2E1 exclusively occurs in transcripts that end in exon T2E3, this relationship is lost in the graph, where E2 can connect to both E5 and E6. As will be discussed later, our approach does not suffer from this shortcoming, since we only extract local information about alternative exon- or intron-usage.

The same principle that was applied when collapsing different isoforms that share the same exons into a graph structure, can be applied again to collapse exonic segments that are shared by several exons/nodes of the splicing graph. Following this idea, we divide each exon into non-overlapping segments. Analog to an exon, a segment is uniquely identified by its coordinate pair and the same order as on exons can be applied: $s = (s_{\text{start}}, s_{\text{stop}})$. We say an exon $v_i$ is *composed* from segments $s_{i,q}$ through $s_{i,r}$, if $v_i = s_{i,q} \circ s_{i,r}$, with $q < r$ and where $\cdot \circ \cdot$ denotes the concatenation of segment positions. Thus, the set of all segments can be defined as

$$S = \bigcup_{v_i \in V} (s_{i,q}, \ldots, s_{i,r} \mid s_{i,q} \circ s_{i,r} = v_i).$$

To explicitly define the set of all segments, at first we define the set $V_{\text{S}}$ of all node-starts in $V$ and the set $V_\mathrm{T}$ of all node stops in $V$. The set of all segments $S$ can then be defined as

$$S = \bigcup_{s_\mathrm{start}, s_\mathrm{stop} \in V_\mathrm{S} \cup V_\mathrm{T}} \{(s_\mathrm{start}, s_\mathrm{stop}) \mid \exists v \in V\colon v_\mathrm{start} \leq s_\mathrm{start} < s_\mathrm{stop} \leq v_\mathrm{stop}\}.$$

The computation of $S$ from $V$ is straightforward. Let $P$ be a sorted array containing all genomic positions that are either start or end of an exon in $V$. We denote the $i$th element of the array as $P[i]$. Let $L_S$ and $L_E$ be two binary label-arrays with the same length as $P$, where $L_S[i]$ is 1 if $P[i]$ is start of an exon in $V$ and 0 otherwise. Analogously, $L_E[i]$ is 1 if $P[i]$ is end of an exon in $V$ and 0 otherwise. Let further $C_S$ and $C_E$ be two arrays with the same length as $P$, where $C_S[i] = \sum_{j=1}^{i} L_S[i]$ and $C_E = \sum_{j=1}^{i} L_E[i]$ are the cumulative starts and ends up to position $i$. We can then determine the set of all segments as

$$S = \bigcup_{i=1}^{|P|-1} \{(P[i], P[i+1]) \mid C_S[i] > C_E[i]\}\,.$$

Analog to the definition of the edges for the splicing graph, we define

$$\begin{aligned} T = \bigcup_{s_u, s_v \in S} \{(s_u, s_v) \mid & \exists v_i \in V, s_r \in S\colon v_i = (s_{r,\mathrm{start}}, s_{u,\mathrm{stop}}) \text{ and} \\ & \exists v_j \in V, s_t \in S\colon v_j = (s_{v,\mathrm{start}}, s_{t,\mathrm{stop}}) \text{ and} \\ & (v_i, v_j) \in E\} \end{aligned}$$

to be the set of segment pairs that are connected by an intron. We then denote the pair $R = (S, T)$ to be the segment graph of a gene. For practical reasons, we store an additional matrix, that relates each node/exon in the splicing graph to the segments it is composed of.

We will use the splicing graph representation to incorporate new information based on RNA-Seq evidence as well as for the extraction of alternative splicing events. However, we will use the segment graph representation for event quantification, as this is computationally much more efficient.

## A.2 Splicing Graph Augmentation

The augmentation of the splicing graph $G$ is a step-wise heuristic. In each step, either a new node or a new edge is added to the graph. If a newly added node shares one boundary with an existing node, the existing edges are inherited by the new node. We will formalize this procedure in the following. We begin by defining the genome $\mathcal{G}$ as a string of consecutive positions $\mathcal{G} = g_1 g_2 \ldots g_n$. Given an RNA-Seq sample and the start $g_s$ and end $g_e$ of a gene, we extract all intron junctions from the alignment, that overlap this region and show sufficient alignment support. Whether an intron junction is sufficiently well supported, is based on a set of given confidence criteria. These criteria will be discussed later in this section. We define the list of RNA-Seq intron junctions $\mathcal{R}$ as

$$\mathcal{R} = \{(g_i, g_j) \mid s \leq i < j \leq e\},$$

where $(g_i, g_j)$ describes the intron starting at $g_i$ and ending at $g_j$. An existing node in the splicing graph $v \in V$ will be represented as the tuple of its genomic coordinates $v = (g_x, g_y)$. If we directly access the coordinate tuple of a node, this is denoted by, $v_\mathrm{start}$ and $v_\mathrm{end}$, thus $v_\mathrm{start} = g_x$ and $v_\mathrm{end} = g_y$. The augmentation process will transform the existing splicing graph $G = (V, E)$ into an augmented version $\hat{G} = (\hat{V}, \hat{E})$. We initialize $\hat{G}$ with $G$.

### A.3 Adding Cassette Exons

In the first round of augmentation, new cassette exon structures are added to the splicing graph. For this, the algorithm iterates over all non-overlapping pairs of $R$. For each pair $(g_{i_1}, g_{j_1})$ and $(g_{i_2}, g_{j_2})$, the following conditions need to be fulfilled, such that a new cassette exon will be added to the graph:

- $\exists v_i \in \hat{V} : v_{i,\text{end}} = g_{i_1} - 1$ and $\exists v_j \in \hat{V} : v_{j,\text{start}} = g_{j_2} + 1$ and $v_i < v_j$
- $\nexists v_h \in \hat{V} : v_{h,\text{start}} = g_{j_1}$ and $v_{h,\text{end}} = g_{i_2}$

Briefly, both introns need to be attached to existing exons and the cassette exon must not already exist. If all conditions are met, a new node $v_n = (g_{j_1} + 1, g_{i_2} - 1)$ is added to the node set $\hat{V}$ and two new edges $(v_i, v_n)$ and $(v_n, v_j)$ are added to $\hat{E}$. Figure 4, Panel A, shows schematically how a cassette exon is added. The criteria for added cassette exons are listed in Table 1.

| **Criterion** | **Value** |
|---|---|
| min exon coverage | 5 |
| min fraction of covered positions in exon | 0.9 |
| min relative coverage difference to flanking exons | 0.05 |

Table 1: Settings for accepted cassette exons

### A.4 Adding Intron Retentions

The second augmentation round adds intron retention events to the splicing graph. For each edge $(v_s, v_t) \in \hat{E}$, the algorithm decides if there is enough evidence from the given RNA-Seq sample for expression inside the intron, to consider the intronic sequence as exonic. Again, heuristic confidence criteria are applied that are listed in Table 2. Briefly, the central criteria for adding a new intron retention are the number of sufficiently covered positions within the intron as well as the differences in mean coverage between intronic and exonic part of that regions. In case of sufficient evidence for a retention, a new node $v_n = (v_{s,\text{start}}, v_{t,\text{end}})$ is added to $\hat{V}$. The new node inherits all incoming edges from $v_s$ and all outgoing edges from $v_t$, thus we get the set of newly added edges

$$E_n = \Big\{(x, v_n) \mid \forall x : (x, v_s) \in \hat{E}\Big\} \cup \Big\{(v_n, x) \mid \forall x : (v_t, x) \in \hat{E}\Big\}.$$

Then, the set of edges is updated with $\hat{E} := \hat{E} \cup E_n$. Figure 4, Panel B, illustrates this case.

| **Criterion** | **confidence level** | | | |
|---|---|---|---|---|
| | **0** | **1** | **2** | **3** |
| min intron cov. | 1 | 2 | 5 | 10 |
| min fraction of cov. positions in intron | 0.75 | 0.75 | 0.9 | 0.9 |
| min intron cov. rel. to flanking exons | 0.1 | 0.1 | 0.2 | 0.2 |
| max intron cov. rel. to flanking exons | 2 | 1.2 | 1.2 | 1.2 |

Table 2: Settings for accepted intron retentions

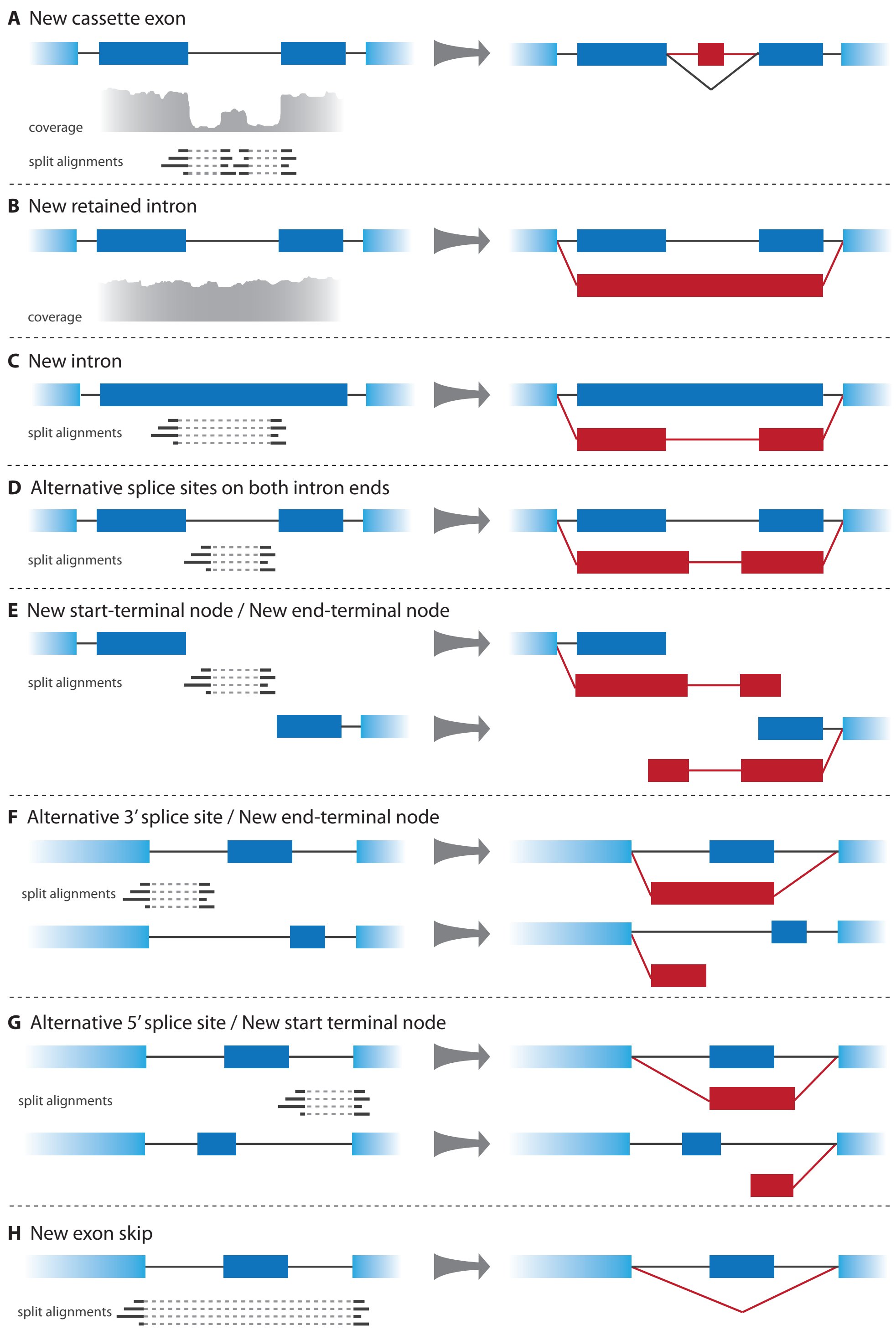
A New cassette exon
coverage
split alignments
B New retained intron
coverage
C New intron
split alignments
D Alternative splice sites on both intron ends
split alignments
E New start-terminal node / New end-terminal node
split alignments
F Alternative 3′ splice site / New end-terminal node
split alignments
G Alternative 5′ splice site / New start terminal node
split alignments
H New exon skip
split alignments

Figure 4: Overview of the different classes of splicing graph augmentation. Panels ***A***–***H*** show all possibilities how the splicing graph can be augmented within SplAdder, based on evidence from RNA-Seq alignment data. In cases where no coverage evidence is shown, only junction confirmations by split alignments are used.

## A.5 Handle Introns

The last augmentation step iterates another time over the list of RNA-Seq supported intron junctions $\mathcal{R}$ that has been generated during the first step. Based on start and end position of the intron, we can test if any existing nodes end or start at these positions, respectively. We have to distinguish four different basic cases: 1) neither start nor stop coincide with any existing node boundary, 2) the intron-start coincides with an existing node end, 3) the intron end coincides with an existing node-start, 4) both the intron-start coincides with an existing node end and the intron-end coincides with an existing node-start. In the following, we will discuss all four cases in further detail. The four cases and their respective sub-cases are illustrated in Figure 4, Panels C–H, which provide a more intuitive explanation and may help the understanding of the following formal definitions.

In the following, we formally define all cases to insert new intron edges into the graph.

1. To handle the first case we can split it into three sub-cases:

   (a) If the intron $(g_i, g_j)$ is fully contained within an existing node ($\exists v \in \hat{V} : g_i > v_{\text{start}}$ and $g_j < v_{\text{end}}$), we can insert a new intron into the node, thus creating two new nodes $v_{n_1} = (v_{\text{start}}, g_i - 1)$ and $v_{n_2} = (g_j + 1, v_{\text{end}})$. After adding $v_{n_1}$ and $v_{n_2}$ to $\hat{V}$, we update the edge set to

$$\begin{aligned}\hat{E} = \hat{E} &\cup \{(v_{n_1}, v_{n_2})\}\\ &\cup \bigcup_{x \in \hat{V}} \left\{(x, v_{n_1}) \mid (x, v) \in \hat{E}\right\}\\ &\cup \bigcup_{x \in \hat{V}} \left\{(v_{n_2}, x) \mid (v, x) \in \hat{E}\right\}\end{aligned}$$

   (b) If the intron $(g_i, g_j)$ is fully contained within an existing intron, we can connect it to the two nodes $v_s$ and $v_t$ flanking the containing intro, thus introducing two new nodes $v_{n_1} = (v_{s,\text{start}}, g_i - 1)$ and $v_{n_2} = (g_j + 1, v_{t,\text{end}})$ into $\hat{V}$. Again, the new nodes inherit their edges from $v_s$ and $v_t$ providing the following update rule for the edge set:

$$\begin{aligned}\hat{E} = \hat{E} &\cup \{(v_{n_1}, v_{n_2})\}\\ &\cup \bigcup_{x \in \hat{V}} \left\{(x, v_{n_1}) \mid (x, v_s) \in \hat{E}\right\}\\ &\cup \bigcup_{x \in \hat{V}} \left\{(v_{n_2}, x) \mid (v_t, x) \in \hat{E}\right\}\end{aligned}$$

   (c) If one of the intron boundaries $(g_i, g_j)$ is in close proximity (we use $\leq 40$ nt as a default threshold) to a terminal node, this node is extended to a new node $v_{n_1}$ and a new terminal node $v_{n_2}$ is added to the graph at the other side of the intron. The length $k$ of the new terminal exon is pre-defined to be 200 nt. If the nearby node $v$ is start-terminal, $v_{n_1} = (g_j + 1, v_{\text{end}})$ and $v_{n_2} = (g_i - k - 1, g_i - 1)$ and

$$\hat{E} = \hat{E} \cup \{(v_{n_2}, v_{n_1})\} \cup \bigcup_{x \in \hat{V}} \left\{(v_{n_1}, x) \mid (v, x) \in \hat{E}\right\}.$$

If the nearby node $v$ is end-terminal, $v_{n_1} = (v_{\text{start}}, g_i - 1)$ and $v_{n_2} = (g_j + 1, g_j + k + 1)$ and

$$\hat{E} = \hat{E} \cup \{(v_{n_1}, v_{n_2})\} \cup \bigcup_{x \in \hat{V}} \left\{(x, v_{n_1}) \mid (x, v) \in \hat{E}\right\}.$$

2. The second case is similar in its handling to case 1c). If the start of intron $(g_i, g_j)$ coincides with the end of an existing node $v$, we can distinguish two sub-cases.

   (a) There exists a node $v'$ in close proximity to intron-end $g_j$ and we can add a new node $v_n = (g_j + 1, v'_{\text{end}})$ and update the edge set to

   $$\hat{E} = \hat{E} \cup \{(v, v_n)\} \cup \bigcup_{x \in \hat{V}} \left\{(v_n, x) \mid (v', x) \in \hat{E}\right\}.$$

   (b) There is no node in close proximity to intron-end $g_j$, thus we introduce a new end-terminal node $v_n = (g_j + 1, g_j + k + 1)$ and update the edge set to $\hat{E} = \hat{E} \cup \{(v, v_n)\}$.

3. The third case is analog to case 2). If the end of intron $(g_i, g_j)$ coincides with the start of an existing node $v$ in the graph, we again can distinguish two sub-cases.

   (a) There exists a node $v'$ in close proximity to $g_i$ and we can add a new node $v_n = (v'_{\text{start}}, g_i - 1)$ and update the edge set to

   $$\hat{E} = \hat{E} \cup \{(v_n, v)\} \cup \bigcup_{x \in \hat{V}} \left\{(x, v_n) \mid (x, v') \in \hat{E}\right\}.$$

   (b) There is no node in close proximity to intron-start $g_i$, thus we introduce a new start-terminal node $v_n = (g_i - k - 1, g_i - 1)$ and update the edge set to $\hat{E} = \hat{E} \cup \{(v_n, v)\}$.

4. The last case is the most straightforward to handle. If intron $(g_i, g_j)$ coincides with the end of node $v$ and the start of node $v'$, we augment the edge set $\hat{E} = \hat{E} \cup \{(v, v')\}$, if the edge is not already present in $\hat{E}$.

## B Extraction of Alternative Splicing Events

Starting with the augmented splicing graph $\hat{G} = (\hat{V}, \hat{E})$, we can extract all alternative splicing events as sub-graphs of the splicing graphs:

**Exon Skips** are all sub-graphs

$$(V', E') = (\{v_i, v_j, v_k\}, \{(v_i, v_j), (v_j, v_k), (v_i, v_k)\})$$

with $V' \subseteq \hat{V}$ and $E' \subseteq \hat{E}$.

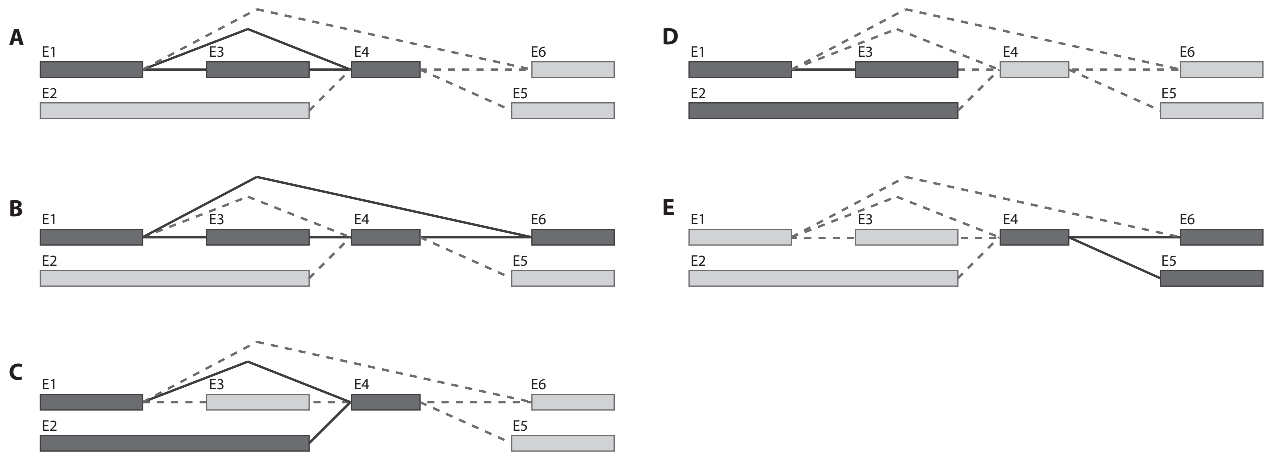

Figure 5: Five different types of alternative splicing events are extracted from the splicing graph. The graph structure is given with nodes as gray boxes and edges as solid/dashed lines. Solid/dark parts show the event of interest and light/dashed parts the remainder of the graph structure. ***A:*** Exon skip, ***B:*** Multiple exon skip, ***C:*** Alternative 5' splice site, ***D:*** Intron retention, ***E:*** Alternative 3' splice site.

**Intron Retentions** are all sub-graphs

$$(V', E') = (\{v_i, v_j, v_k\}, \{(v_i, v_j)\})$$

with $V' \subseteq \hat{V}$ and $E' \subseteq \hat{E}$ and $v_{i,\text{start}} = v_{k,\text{start}}$ and $v_{j,\text{end}} = v_{k,\text{end}}$.

**Alternative 3' Splice Sites** are all sub-graphs

$$(V', E') = (\{v_i, v_j, v_k\}, \{(v_i, v_j), (v_i, v_k)\})$$

with $V' \subseteq \hat{V}$ and $E' \subseteq \hat{E}$ and $v_{j,\text{end}} = v_{k,\text{end}}$. This definition assumes the direction of transcription to be positive. For transcripts from the negative strand, the definition for alternative 3' splice site and alternative 5' splice site need to be switched.

**Alternative 5' Splice Sites** are all sub-graphs

$$(V', E') = (\{v_i, v_j, v_k\}, \{(v_i, v_k), (v_j, v_k)\})$$

with $V' \subseteq \hat{V}$ and $E' \subseteq \hat{E}$ and $v_{i,\text{start}} = v_{j,\text{start}}$. The different strands are handled analogously to alternative 3'-splice sites.

**Multiple Exon Skips** are all sub-graphs

$$\begin{aligned}(V', E') =& (\{v_i, v_{j_1}, \ldots, v_{j_s}, v_k\}, \{(v_i, v_{j_1}), (v_{j_s}, v_k), (v_i, v_k)\} \\ & \cup \bigcup_{l=1}^{s-1} \{(v_{j_l}, v_{j_{l+1}})\}\})\end{aligned}$$

with $V' \subseteq \hat{V}$ and $E' \subseteq \hat{E}$.

The same extraction rules would apply analogously, to extract alternative splicing events from the not augmented graph $G$. A schematic overview of the extraction process is provided in Figure 5.

### B.1 Event Filtering and Quantification

Alternative splicing events extracted from the graph are filtered at several levels. To remove redundant events, all events are made unique based on their inner event coordinates. The inner event coordinates are defined as the start and end positions of all introns of the event. If two events share the same inner coordinates, they are replaced by a new event with the same inner coordinates but adapted outer coordinates minimizing the total length of the event. An example for this is shown in Figure 6. Events in Panel A can be merged, whereas events in Panel B disagree in their inner coordinates and remain separate.

In the next step we use the RNA-Seq data to quantify each of the extracted events. That is, for each intron we count the number of alignments supporting it and compute the mean coverage for each exon. For reasons of computational efficiency, the quantification is performed on the segment graph. As defined above, each segment can be uniquely identified by its genomic coordinates. Thus, we extract for each node its mean coverage and for each edge the number of spliced alignments in the sample confirming this edge. As each exon $v_i$ can be formed through a concatenation of segments $s_q \circ s_r$, we can use the segment-lengths and their average coverage to compute the average coverage of the exon:

$$v_{i,\text{coverage}} = \frac{\sum_{j=q}^{r}(s_{j,\text{stop}} - s_{j,\text{start}} + 1) \cdot s_{j,\text{coverage}}}{\sum_{j=q}^{r}(s_{j,\text{stop}} - s_{j,\text{start}} + 1)},$$

where $s_q \circ s_r$ is the sequence of segments contained in node $v_i$.

In many applications, the splicing graphs can grow very complex, containing alternative events that are only poorly supported by input data. Thus, we use the quantifications to further filter the event set and to only retain the most confident events. Each event type has a different set of criteria it has to fulfill in order to become a valid event. Complete listings of the respective criteria are provided in Table 3. To determine, if an event is valid, the algorithm checks in which provided RNA-Seq samples which criteria are met. An event is valid, if all criteria are met in at least one sample. To create more stringently filtered sets of events, this threshold can be increased.

| **Criterion** | **Confidence Level** | | | |
|---|---|---|---|---|
| | **0** | **1** | **2** | **3** |
| min segment length | $\lceil 0.1 \cdot r \rceil$ | $\lceil 0.15 \cdot r \rceil$ | $\lceil 0.2 \cdot r \rceil$ | $\lceil 0.25 \cdot r \rceil$ |
| max mismatches | $\max\{2, \lfloor 0.03 \cdot r \rfloor\}$ | $\max\{1, \lfloor 0.02 \cdot r \rfloor\}$ | $\max\{1, \lfloor 0.01 \cdot r \rfloor\}$ | 0 |
| max intron length | 350,000 | 350,000 | 350,000 | 350,000 |
| min junction count | 1 | 2 | 2 | 2 |

Table 3: Settings for accepted introns

## C Evaluation and Testing

The SplAdder software has been developed in the context of application and has been successfully used in numerous projects on *Arabidopsis thaliana* [7, 1, 3] as well as in large scale cancer projects [9].

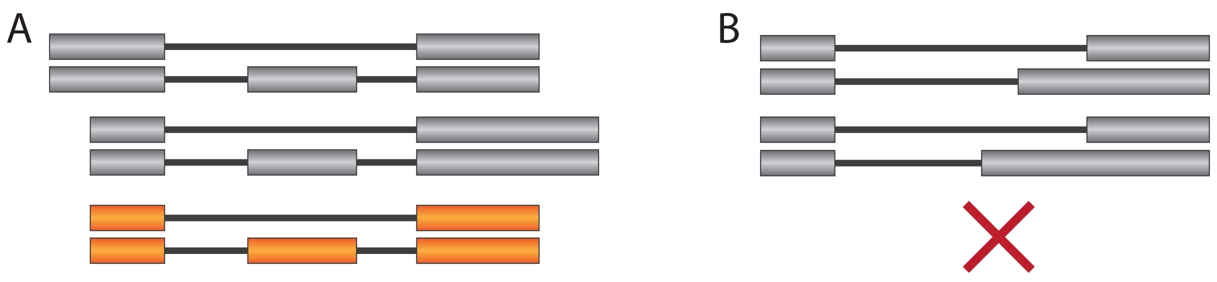

Figure 6: Example case when overlapping events can be merged. ***A:*** All inner event coordinates agree and the events can be successfully merged. ***B:*** Both events have only one intron in common, whereas the other introns disagree. The events cannot be merged and remain separate.

However, to allow for an accurate measure of performance, we have used simulated data in this work to assess the SplAdder results.

We used the FluxSimulator (in its version 1.1.1-20121103021450) [4] to simulate RNA-Seq reads generated based on 5,000 genes randomly selected from the human reference annotation (Gencode, NCBI36). The read simulation produced 76bp paired-end reads using the following parameters:

```
EXPRESSION_X0       9500
EXPRESSION_K        -0.6
TSS_MEAN            50
POLYA_SCALE         300
POLYA_SHAPE         2
FRAG_SUBSTRATE      DNA
FRAG_METHOD         NB
FRAG_NB_LAMBDA      500
FILTERING           YES
SIZE_DISTRIBUTION   N-300-50.txt
SIZE_SAMPLING       AC
RTRANSCRIPTION      YES
PCR_PROBABILITY     0.7
RT_PRIMER           PDT
RT_LOSSLESS         YES
RT_MIN              500
RT_MAX              5500
PAIRED_END          YES
FASTA               YES
```

Where `N-300-50.txt` contains a random sample, drawn from a normal distribution with mean 300 and standard deviation 50.

In total, we generated 20,000,000 reads from 5,000 genes containing 11,591 transcript isoforms. These reads were aligned back to the hg19 reference genome sequence using *PALMapper* [5]. For the purpose of this evaluation, we removed all but the first isoform from each gene and stored this as a *backbone annotation*, which was provided to SplAdder along with the simulated reads. To assess how much complexity could be restored by SplAdder, we compared the splicing graph built on the original annotation to the ones that had been created from the SplAdder augmented backbone annotation. We overlapped three categories: the introns (edges in the graph), the exons (nodes in the graph) and the inner exons (non-terminal nodes in the graph). The overlap evaluation is summarized in Figure 7. Whereas the precision (middle panel) increases for all graph elements that have been evaluated with increasing confidence level, the recall (left panel) slightly drops as input data is more stringently filtered. The F-measure (right panel), computed as the harmonic mean of precision and recall, is close to 0.8 for all confidence levels and peaks at confidence level 1.

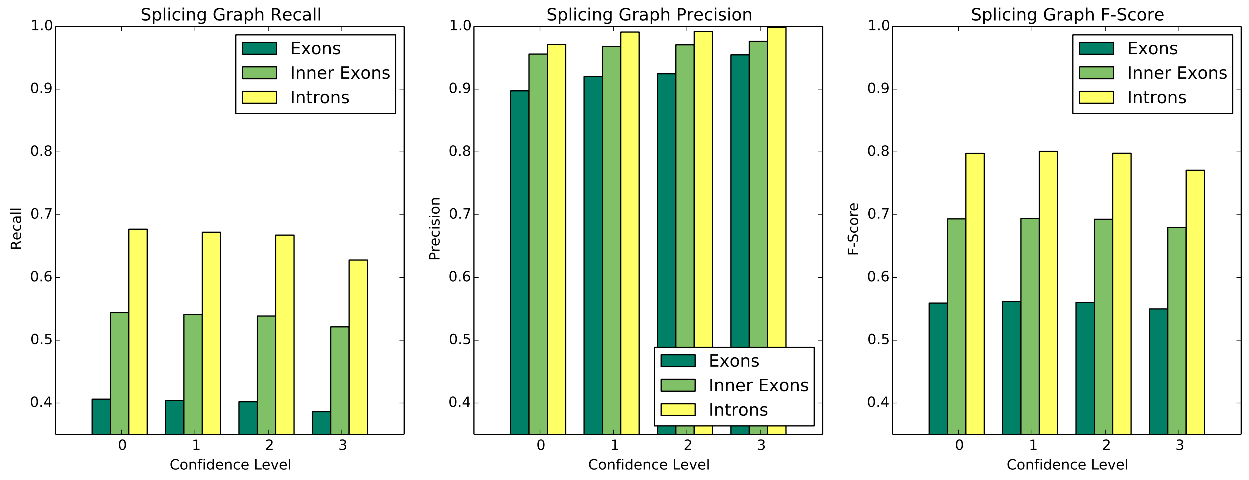

Figure 7: Results of the performance evaluation based on simulated read data. Precision, recall and F-measure, shown left, middle and right, respectively, were computed for nodes and edges of the graph for the three different SplAdder confidence levels.

## D Visualizations

Being able to transform the large amount of splicing information available for a gene locus is an important step towards a better understanding of altered splicing mechanisms or to identify impaired RNA regulation. To aid with this, SplAdder is able to produce a variety of diagnose and overview-plots to summarize information at a specific locus or to given an overview on the distributions of certain characteristics of all identified events.

An illustrative example is the gene-locus overview plot that can summarize the splicing graph of a gene and align it to the coverage in a set of given samples, thereby highlighting coverage differences (cf. Figure 8).

The list of available plotting routines is constantly extended. Please refer to the user documentation for a more comprehensive overview.

## References

[1] Gabriele Drechsel, André Kahles, Anil K Kesarwani, Eva Stauffer, Jonas Behr, Philipp Drewe, Gunnar Rätsch, and Andreas Wachter. Nonsense-Mediated Decay of Alternative Precursor mRNA Splicing Variants Is a Major Determinant of the Arabidopsis Steady State Transcriptome. *The Plant Cell*, 25(10):3726–3742, 2013.

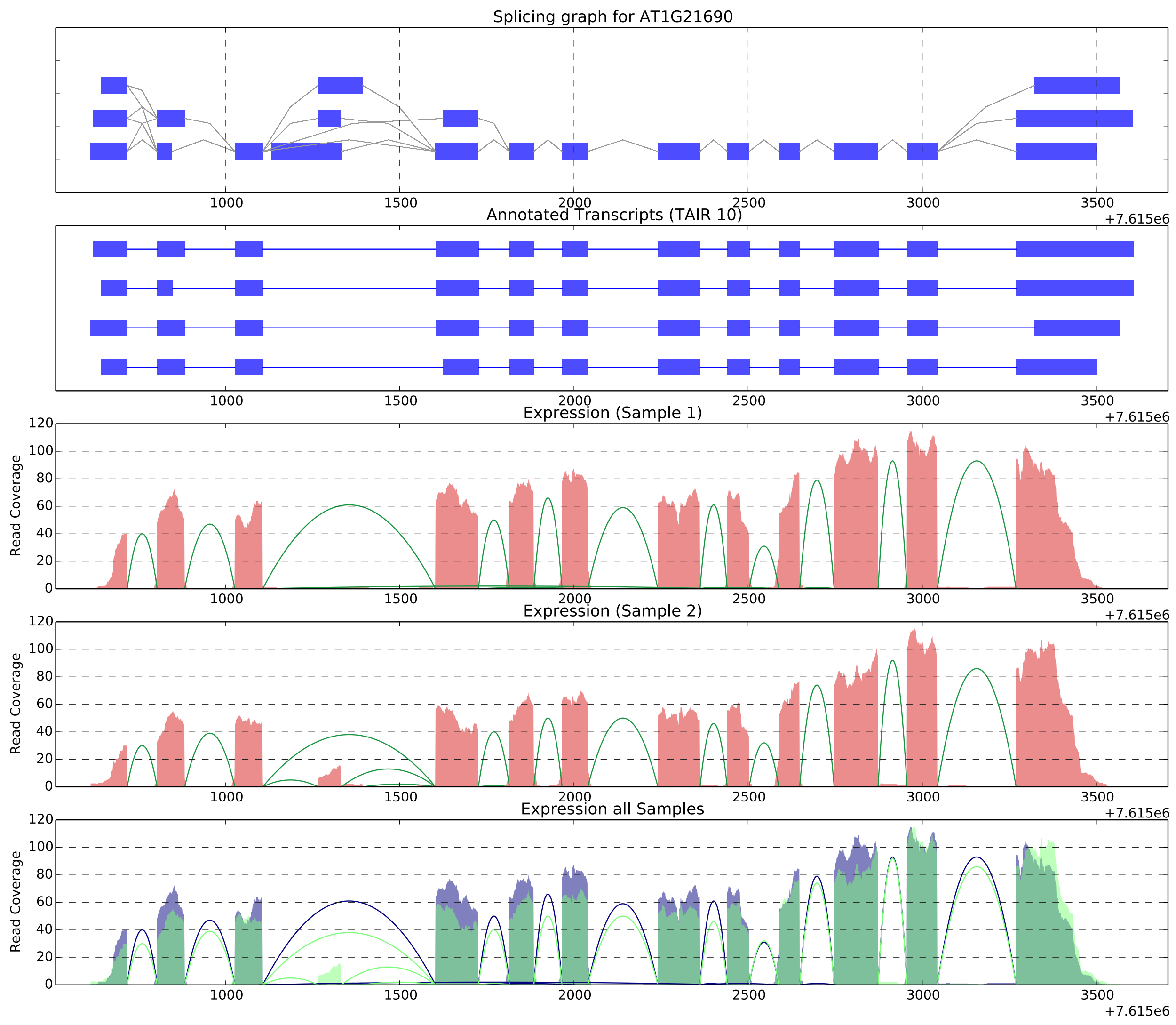

Figure 8: Visualization of the splicing pattern occurring at a certain gene locus. The example shows real data taken from the experiments on *Arabidopsis thaliana* NMD impaired mutants published in [1]. The upper track shows the splicing graph for the gene AT1G21690 that has been generated by SplAdder. The second track shows the annotated transcripts isoforms available in the TAIR10 annotation. Note, that none of the transcripts contains an additional exon identified by SplAdder. When looking at the coverage overview in the WT (track 3) and double-knockdown (track 4) samples, a clear differential usage of that novel exon is apparent. Lastly, track 5 shows both samples in a comparative manner.

[2] Philipp Drewe, Oliver Stegle, Lisa Hartmann, André Kahles, Regina Bohnert, Andreas Wachter, Karsten Borgwardt, and Gunnar Rätsch. Accurate detection of differential RNA processing. *Nucleic Acids Research*, 41(10):5189–5198, May 2013.

[3] X Gan, O Stegle, J Behr, J G Steffen, P Drewe, K L Hildebrand, R Lyngsoe, S J Schultheiss, E J Osborne, V T Sreedharan, A Kahles, R Bohnert, G Jean, P Derwent, P Kersey, E J Belfield, N P Harberd, E Kemen, C Toomajian, P X Kover, R M Clark, G Rätsch, and R Mott. Multiple

reference genomes and transcriptomes for Arabidopsis thaliana. *Nature*, 108(25):10249–10254, August 2011.

[4] Thasso Griebel, Benedikt Zacher, Paolo Ribeca, Emanuele Raineri, Vincent Lacroix, Roderic Guigó, and Michael Sammeth. Modelling and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic Acids Research*, 40(20):10073–10083, September 2012.

[5] G Jean, A Kahles, V T Sreedharan, F De Bona, and G Rätsch. RNA-Seq read alignments with PALMapper. *Current Protocols in Bioinformatics*, Chapter 11(December):Unit 11.6, December 2010.

[6] 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(12):1413–1415, 2008.

[7] C Rühl, E Stauffer, A Kahles, G Wagner, G Drechsel, G Rätsch, and A Wachter. Polypyrimidine Tract Binding Protein Homologs from Arabidopsis Are Key Regulators of Alternative Splicing with Implications in Fundamental Developmental Processes. *The Plant Cell*, 24(11):4360–4375, 2012.

[8] Eric T Wang, Rickard Sandberg, Shujun Luo, Irina Khrebtukova, Lu Zhang, Christine Mayr, Stephen F Kingsmore, Gary P Schroth, and Christopher B Burge. Alternative isoform regulation in human tissue transcriptomes. *Nature*, 456(7221):470–476, 2008.

[9] John N Weinstein, Eric a Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad a Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, and Joshua M Stuart. The Cancer Genome Atlas Pan-Cancer analysis project. *Nature Genetics*, 45(10):1113–1120, October 2013.