LEHRSTUHL FÜR EXPERIMENTELLE BIOINFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Final Report of the Fortgeschrittenen Praktikum

Evaluation of differential Splicing tools

Alexander Dietrich

Dr. Markus List, Dr. Olga Tsoy, Prof. Dr. Jan Baumbach

Contents

1 Introduction

1.1 Alternative Splicing

Splicing describes the post-transcriptional editing of eukaryotic mRNA. The results of Alternative Splicing are different isoforms of the mRNA for a single gene, an isoform being a distinct selection of coding sequence regions, the exons. These different isoforms lead to different proteins with potentially different functions encoded by the same gene (figure 1.1). Alternative Splicing can appear in many different types and combinations of events, which will be explained in detail in the next chapter.
As a result the number of different proteins in a mammal can exceed the number of genes by a factor of four [1], which shows the importance of alternative splicing regarding protein diversity. Also many diseases are presumed to be linked to irregularities in this complex process [2]. All the different mechanisms, which influence alternative splicing regulation, are still far from being fully understood [3].
Alternative splicing detection tools aim to identify splicing events by utilizing RNA-seq data.

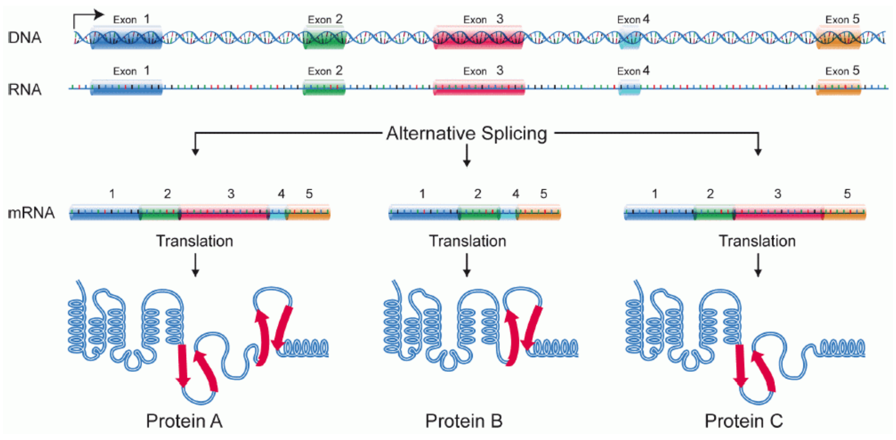

Figure 1.1: Alternative splicing enables a single gene to encode multiple proteins [4].

1.1.1 Alternative Splicing Event types

There are seven types of alternative splicing in human: exon skipping (ES), intron retention (IR), alternative 5′/donor splice site (A5), alternative 3′/aceptor splice site (A3), mutually exclusive exons (MEE), multiple exon skipping (MES), alternative first exon (AFE) and alternative last exon (ALE) (figure 1.2).
Exon skipping describes the event, when a single exons surrounded by 2 other exons is removed by splicing, resulting in two separate exons being joined together; multiple exon skipping is the same principle, only more than one exon is removed. Intron retention occurs, when a single intron is not spliced out. Alternative 3′ and 5′ splice site describes the event, in which the position of the splice site at the 3′ or 5′ end of an exon is changed. When two ES events are not independent anymore, but rather are executed in coordination (one ES event is only performed, if the other event is not performed), this is called mutually exclusive exons. Alternative last and first exon simply is the case, when the first or last exon of a gene is spliced in or not.

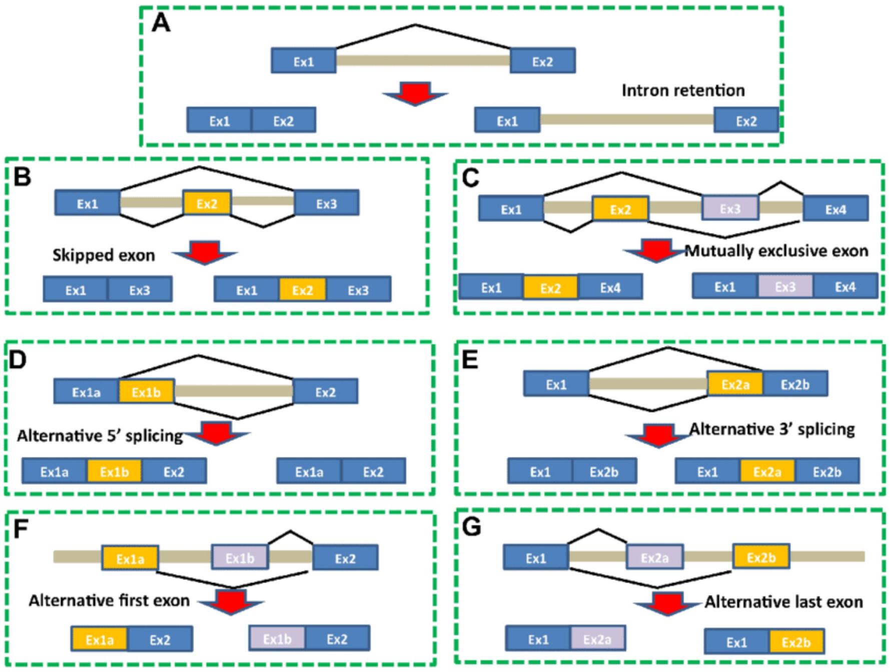

Figure 1.2: Schematic of different alternative splicing event types [5]

1.2 Bioinformatic Tools to detect Splicing Events

There are several bioinformatic tools, which can detect those different event types. During this project four of them were used, each with a different output of detected events and with different file formats. The standard approach here is to align short RNA-seq reads to a reference genome, followed by comparing these alignments to a given genome annotation. The genome annotation is used to build a (directed) splice-graph, were nodes represent the exons, edges the different splice junctions. A path through this graph is then considered as a valid isoform. Figure 1.2 B shows such a splice graph for a single exon skipping event.
See the four used tools, their main programming language and version in table 1.1.

name	language	version
MAJIQ	python	XXX
SplAdder	python	XXX
ASGAL	python/C++	XXX
Whippet	julia	XXX

Table 1.1: Alternative Splicing event detection tools

1.3 Motivation

Initially each of the above mentioned tools aims to do the same overall thing: finding alternative splicing events; but each tool has its own approach and therefore its own way of generating output files for the found events and annotating an event. This makes it really hard for a user to compare multiple tools. The goal of project was to create an easy to use tool, which can handle those different tool outputs and create a single new file for each, without loosing information. The new file will then have the same structure for all alternative splicing event detection tools, making precise comparisons per gene or on coordinate level possible.

2 Methods

2.1 Unifying outputs of Alternative Splicing event detection tools into a single format

In order to achieve easy comparison between tools, one file-format was decided on. A tab separated file will store every event, one tool finds, each event encoded by one of the seven explained standard event types. Additionally, for each event the gene name, chromosome, strand as well as a unique ID (usually the ID the tool already gives, with some minor additions in some cases) will be listed. The most important part of the file are the genome coordinates for each event. The following table 2.1 will explain in detail, which coordinates are reported for each event type. Event types with more than one start and stop coordinate (MES and MEE), a comma separated list of start or stop coordinates will be given.

event-type	strand	start-coordinates	end-coordinates
ES	+/-	start of skipped exon	end of skipped exon
IR	+/-	start of retained intron (previous exon-end +1)	end of retained intron (next exon -1)
A5	+ -	alternative exon-end regular exon-start	regular exon-end alternative exon-start
A3	+ -	regular exon start alternative exon end	alternative exon start regular exon end
AFE	+/-	start of alternative first exon	end of alternative first exon
ALE	+/-	start of alternative last exon	end of alternative last exon
MEE	+/-	start of exclusive exon 1, start of exon exclusive exon 2	end of exclusive exon 1 end of exclusive exon2
MFE	+/-	start of skipped exon 1, start of exon skipped exon 2, ..., start of exon skipped exon n	end of skipped exon1 end of skipped exon 2, ..., end of skipped exon n

Table 2.1: genome coordinates for each event type

Note that for A3 and A5 events, the strand has to be considered as well. This difference is also highlighted in figure 2.1.

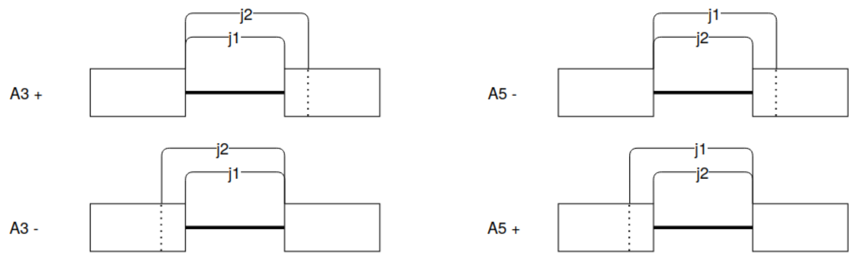

Figure 2.1: schematic view of A3 and A5 events for both strands; + means the strand from 5′ to 3′, - means from 3′ to 5′. Boxes represent exons, bold horizontal lines the intron in between. The dashed line represents the alternative splice junction, j1 and j2 represent the 2 possible splice junctions.

The next sections will each explain how the OUTPUT_TRANSFORMER re-calculates the output of each tool into the proposed unified file format. The exact specifications for each column can be found in the supplementary table 4.1.

2.1.1 MAJIQ

MAJIQ [6] is a software package in the python programming language to detect splicing events. They use a more flexible definition of these events in the form of "local splicing variations" (LSVs), which describes exons, from which (or to which) splicing events start (or end). This means in their definition, a LSV can contain several combinations of splicing events. A binary LSV for example would then simply be a single exon skipping event (an example visualization of such a case can be seen in supplementary figure 4.1). More complex LSVs can contain multiple "standard" event types.

The MAJIQ project also includes the VOILA tool, which is a way of generating visualizations of splicing variations as well as providing a more detailed output format than MAJIQ does. The MAJIQ output file (*MAJIQ.psi*) has columns for each LSV with the coordinates of the corresponding splice junctions as well as three columns named A5SS, A3SS, ES, which have a TRUE/FALSE value, indicating which of those three types are included in the LSV. One additional column named IR_COORDS gives the coordinates of retained introns. To correctly find out all "standard" events in one LSV, additional information about the included exons in a event were needed; ES events for example can not be correctly annotated with only the junction coordinates (4.1).

The MAJIQ tool comes with an additional program called VOILA. Usually this is used to visualize the output of MAJIQ, but it also has the VOILA TSV-command, which creates a more detailed output file for MAJIQ. This includes, among others, a new column called *exon_coords*, which contains the start and end coordinates of each exon contained in the LSV. When starting this project, the goal was to transform the output of the VOILA TSV command into the unified format, but an update of VOILA during production changed the produced output file. This update removed the three columns A5SS, A3SS, ES from the VOILA TSV output. This issue was solved by merging the two output files (MAJIQ and VOILA) together, keeping only the needed columns (id, gene, strand, seqid, junction_coords, exon_coords, ir_coords, A3SS, A5SS and ES) per event. So now the OUTPUT_TRANSFORMER needs both the output file of MAJIQ and VOILA in one directory in order to work. With them all of the following event types could be calculated: ES, A3, A5, IR, MES. MES events are reported if the OUTPUT_TRANSFORMER finds more than one concurrent ES event.
MAJIQ sometimes reports "nan" coordinates, these were just adopted.

2.1.2 SplAdder

SplAdder [7] is a python package, which analysis alternative splicing based on RNA-seq data. They stick the standard type annotation of splicing events as described in 1.1.1, but do not report any AFE or ALE events. SplAdder creates one file per event type. The important columns of these files are: exon_alt1_start, exon_alt1_end, exon_alt2_start and exon_alt2_end; each describes the start/end of one alternative exon. With these columns, all five remaining standard event types can be created. Only for A3/A5 that depends on the current strand, where A3 events on the positive strand will have exon_alt2_start as the start coordinate and exon_alt1_start as the end and A3 events in the negative strand use exon_alt1_end and exon_alt2_end respectively (A5 events are the same just with inverted strands).

2.1.3 ASGAL

The "Alternative Splicing Graph ALigner" - ASGAL [8] is "a tool for mapping RNA-seq data to a splicing graph with the goal of detecting novel splicing events [...]". It reports a comma separated file with one splicing event per line, giving the type, start, end, read support and gene name of this event. The coordinates in this case correspond to the start and end of a alternative junction (this would be the "larger" junction in each of the examples from figure 2.1). The OUTPUT_TRANSFORMER requires more information though, so in this case OUTPUT_TRANSFORMER also requires the used annotation file for ASGAL in gtf format. Even with that additional file, it was only possible to calculate

IR, ES, MES, A3 and A5 events.
For IR events, the start and end coordinate could just be adopted; in order to find skipped exons, the gtf-file was parsed to find all exons in the gene of this event. Then those exons which are within the start and end coordinate, are considered skipped exons. If more than one is found, the event is annotated as MES. For A3 and A5 events a sliding window over the exons was used, always looking at two consecutive exons. Because only the junction coordinates are given by ASGAL, the regular start/end of an A3/A5 event can be calculated with the exon coordinates from the gtf-file. In some cases the start of the junction and the end of the first exon (or the junction end and the start of the second exon) are the same; in this case the start coordinate of the A3/A5 event will be set to "nan" (or the end coordinate). See the detailed pseudo code in supplementary figure 24.
Since ASGAL provides no unique ID for am event, the OUTPUT_TRANSFORMER creates a new ID for each event, consisting of the gene name, start and end coordinate.

2.1.4 Whippet

Whippet [9] is a tool in the julia programming language to model and quantify alternative splicing events. Whippet generates one output *.psi* file, which has three important columns for OUTPUT_TRANSFORMER: the *Type*, the *Exc_Paths* and the *Edges* column. The first one describes for each line the event type (can be NA), for which Whippet uses a different naming convention than the described "standard" events in 1 (see supplementary table 4.2 for a translation; Whippet has some more detailed types, but they did not occur at all during testing Whippet).

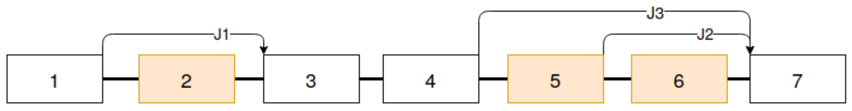

Figure 2.2: node representation in Whippet: each box represents on node in the splice graph. The horizontal bold lines are edges of type n to n+1. Out of J1, J2 and J3 only J3 is considered as a "long_junction". In this case OUTPUT_TRANSFORMER will generate one ES event with the coordinates of box 2 as skipped exon, and one MES event with the coordinates of box 5 and 6 as skipped exons.

For each node in the splice graph of a gene, Whippet creates one line in its output file; each node then gets an event type (or NA).The OUTPUT_TRANSFORMER first scans all lines until a new gene appears and then handles all events it found in the previous lines. ES events are only counted as correct if the Whippet type is CE and the *Exc_Paths* column is not "NA", since this column contains the paths quantified through the AS event. The *Edges* column describes all edges in the splice graph, which go through this node. The average edge just goes from node n to n+1, describing a junction of two consecutive exons. If however an edge covers more than 2 nodes, this edge is saved internally by OUTPUT_TRANSFORMER as a "long_junctions". After processing each line for a gene, each of those "long_junctions" will be checked for any skipped exons, which lie inside of it (see figure 2.2). All other event types (IR, A3, A5, ALE, AFE) did not require any recalculations; Whippet reports the coordinates in the same format as the OUTPUT_TRANSFORMER needs it.

2.2 Alternative approach to handle skipped exons

As the last chapter showed, it was not possible to get all seven event types out of each tool. Especially the MES and MEE events proposed a challenge and the different outputs made it sometimes more or less hard to get the correct coordinates. Still these coordinates might not be very precise in every case, which is why the OUTPUT_TRANSFORMER has the option to combine all MES, MEE and ES events into a single event type: ES.
The approach is quite trivial, for each skipped exon of the MES and MEE events (n and 2 respectively) a single new ES event will be created. So for example one MES event with 5 skipped exons will result in 5 separate ES events. The *count* column will be used to keep track of how many new events were created from one MES/MEE event (for MEE events there are always only two new events). So again for the previous example, the five ES events would get the following entries in the *count* column: 1, 2, 3, 4, 5.
This feature can be turned off and on easily with a flag in the command line (see section 2.4 for all possible flags).

2.3 Comparison with simulated data

Since the overall goal of this project was to allow for easier comparison between tools, a second run mode was implemented, which allows the user to compare the new unified tool output to an event annotation file, where all known events are stored in. For each of the seven event types, this mode will count how many events are correct, by comparing an event of the tool with every event on the same gene with the same type in the annotation. If the "strict" flag is used, for this event the start and end coordinate have to be exactly equal; with this flag, only one of them has to be identical. This can

be useful when a tool creates many "nan" values for example.
The program also has the option to set a threshold value for the minimum allowed distance between two events ($distance_{e1,e2} = |e1.start - e2.start| + |e1.end - e2.end|$). Per default this is set to 0, but can be increased to potentially label more events as correct. Two evaluation scores are calculated: precision and recall. Precision is described as the fraction of correct events divided by the overall number of found events by the tool and recall as the fraction of correct events divided by the number of all events in the annotation. Of course these values are calculated for each event type separately. The raw values as well as the scores can then either be saved in a file or just printed to std-out.

2.4 Documentation and Availability

The OUTPUT_TRANSFORMER has two separate run modes, one to create the unified output file for one (or more tools) and another one to compare this output to an annotation file.

Mode one starts with

```
usage:  output_transformer.py create [-h] [-m MAJIQ_DIR] -s SPLADDER_DIR][-w
WHIPPET_FILE] [-a ASGAL_FILE] -out OUTDIR -gtf GTF [-comb COMBINE_ME]:

  optional arguments:

    -h     -help          show help message and exit
    -s     -spladder_dir  directory of SplAdder output:
                          only confirmed.txt files
    -w     -whippet_file  whippet out.psi file
    -a     -asgal_file    ASGAL.csv out file
    -out   -outdir        output directory
    -gtf   -gtf           reference file in gtf format
    -comb  -combine_me    Set this to true if you want MES and MEE
                          to be counted as ES events
                          (each skipped exon is one separate ES event)
```

Table 2.2: possible command line flags for the create runmode

Mode two starts with

```
usage: output_transformer.py compare [-h] -a EVENT_ANNOTATION -c COMPARE_FILE
-gtf GTF [-stats STATS_OUTFILE] [-comb COMBINE_ME] [-s STRICT] [-t THRESHOLD]:
```

`optional arguments:`

`-h`	`-help`	`show help message and exit`
`-a`	`-event_annotation`	`Event annotation file for ground truth of events only confirmed.txt files`
`-c`	`-compare_file`	`unified output of AS tool that will be checked`
`-comb`	`-combine_me`	`Set this to true if you want MES and MEE events to be counted as ES events (each skipped exon is one separate ES event); should also been used when creating the compare file!`
`-s`	`-strict`	`Use this flag if you want strict comparison between the output and event annotation. Strict means that both start and end coordinate have to be equal so that an event is counted as correct.`
`-t`	`-threshold`	`set threshold to allow for events with minimum distance < threshold to still be counted as correct; default is 0`

Table 2.3: possible command line flags for the compare runmode

The code is currently available as a part of a bigger alternative splicing evaluation project on gitlab: `https://gitlab.lrz.de/ge46ban/dockers`.

3 Results

3.1 Proposing a standard file-format for Alternative Splicing events

3.2 Performance of selected Alternative Splicing tools

4 Discussion

4.1 Possible extensions and applications

4.2 Potential downfalls of file format

Supplementary

column	input
chr	symbol of chromosome for this event
gene	gene name for this event
id	unique identifier for this event
strand	$+$ or $-$
event_type	one of the following types: ES, IR, A3, A5, ALE, AFE, MEE, MES
count	default=1; can be used for tools like MAJIQ, which report multiple events for one ID to keep track of the number of events; *count* in combination with *id* has to be unique
start_coordinates	one or more start coordinates
end_coordinates	one or more end coordinates

Table 4.1: Allowed inputs for each column in the unified output file

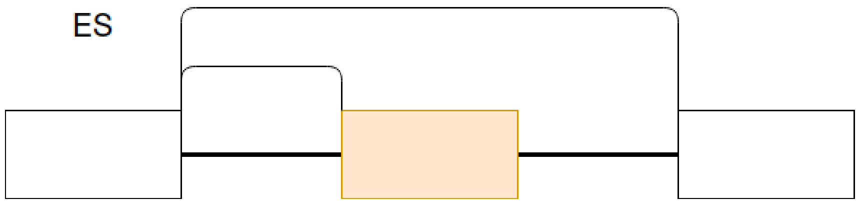

Figure 4.1: visualization of a MAJIQ LSV with a single skipped exon; the two lines show the junction coordinates given by the MAJIQ output. This shows that an exact annotation of the skipped exon (orange box) is not possible with the junction coordinates, at least one coordinate will be missing.

Algorithm 1 Calculate A3 and A5

```
procedure CALCA3_A5(j_start, j_end)                ▷ The start and end of the junction
    for exon1, exon2 do
        if (event_type == A3 and strand == -) or (event_type == A5 and strand == +) then
            if j_end +1 == exon2.start then
                alt_part = (exon1.end, j_start)
                if j_start < exon1.end then        ▷ Check if junction not inside of exon
                    alt_part = [j_start, exon1.end]
                if j_start == exon1.end then
                    alt_part = ["nan", exon1.end]
                if event_type == A3 then
                    return A3_event(alt_part[0], alt_part[1], -)
                else
                    return A5_event(alt_part[0], alt_part[1], +)
        else
            if j_start -1 == exon1.end then
                alt_part = (j_end, exon2.start)
                if exon2.start < j_end then        ▷ Check if junction not inside of exon
                    alt_part = [exon2.start, j_end]
                if j_end == exon2.start then
                    alt_part = [exon2.start, "nan"]
                if event_type == A3 then
                    return A3_event(alt_part[0], alt_part[1], +)
                else
                    return A5_event(alt_part[0], alt_part[1], -)
```

Whippet type	standard type
CE	ES
AA	A3
AD	A5
RI	IR
TS	-
TE	-
AF	AFE
AL	ALE
BS	-

Table 4.2: Whippet event type to standard event type translation

List of Figures

List of Tables

Bibliography

[1] T. W. Nilsen and B. R. Graveley. “Expansion of the eukaryotic proteome by alternative splicing.” In: *Nature* 463.7280 (Jan. 2010), pp. 457–463. ISSN: 1476-4687. DOI: 10.1038/nature08909. URL: https://doi.org/10.1038/nature08909.

[2] R. K. Singh and T. A. Cooper. “Pre-mRNA splicing in disease and therapeutics.” eng. In: *Trends in molecular medicine* 18.8 (Aug. 2012). S1471-4914(12)00101-3[PII], pp. 472–482. ISSN: 1471-499X. DOI: 10.1016/j.molmed.2012.06.006. URL: https://doi.org/10.1016/j.molmed.2012.06.006.

[3] Y. Wang, J. Liu, B. O. Huang, Y.-M. Xu, J. Li, L.-F. Huang, J. Lin, J. Zhang, Q.-H. Min, W.-M. Yang, and X.-Z. Wang. “Mechanism of alternative splicing and its regulation.” eng. In: *Biomedical reports* 3.2 (Mar. 2015). br-03-02-0152[PII], pp. 152–158. ISSN: 2049-9434. DOI: 10.3892/br.2014.407. URL: https://doi.org/10.3892/br.2014.407.

[4] Wikipedia. *Alternative splicing — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Alternative%20splicing&oldid=962571544. [Online; accessed 30-August-2020]. 2020.

[5] N. A. Faustino. “Pre-mRNA splicing and human disease.” In: *Genes & Development* 17.4 (Feb. 2003), pp. 419–437. DOI: 10.1101/gad.1048803. URL: https://doi.org/10.1101/gad.1048803.

[6] J. Vaquero-Garcia, A. Barrera, M. R. Gazzara, J. González-Vallinas, N. F. Lahens, J. B. Hogenesch, K. W. Lynch, and Y. Barash. “A new view of transcriptome complexity and regulation through the lens of local splicing variations.” In: *eLife* 5 (Feb. 2016). Ed. by J. Valcárcel, e11752. ISSN: 2050-084X. DOI: 10.7554/eLife.11752. URL: https://doi.org/10.7554/eLife.11752.

[7] A. Kahles, C. S. Ong, Y. Zhong, and G. Rätsch. “SplAdder: identification, quantification and testing of alternative splicing events from RNA-Seq data.” In: *Bioinformatics* 32.12 (Feb. 2016), pp. 1840–1847. DOI: 10.1093/bioinformatics/btw076. URL: https://doi.org/10.1093/bioinformatics/btw076.

[8] L. Denti, R. Rizzi, S. Beretta, G. D. Vedova, M. Previtali, and P. Bonizzoni. "ASGAL: aligning RNA-Seq data to a splicing graph to detect novel alternative splicing events." In: *BMC Bioinformatics* 19.1 (Nov. 2018). DOI: 10.1186/s12859-018-2436-3. URL: https://doi.org/10.1186/s12859-018-2436-3.

[9] T. Sterne-Weiler, R. J. Weatheritt, A. J. Best, K. C. Ha, and B. J. Blencowe. "Efficient and Accurate Quantitative Profiling of Alternative Splicing Patterns of Any Complexity on a Laptop." In: *Molecular Cell* 72.1 (Oct. 2018), 187–200.e6. DOI: 10.1016/j.molcel.2018.08.018. URL: https://doi.org/10.1016/j.molcel.2018.08.018.