User Guide for SplicingTypesAnno Package

Xiaoyong Sun[†*], Fenghua Zuo[‡]

August 7, 2012

[†]McDermott Center for Human Growth & Development
The University of Texas Southwestern Medical Center
Dallas, TX 75390, USA
[‡]Department of Information Science
Taishan Medical University
Taian, Shandong, China

Contents

*johnsunx1@gmail.com

1 Introduction

Alternative splicing plays a key role in the central dogma. Alternative splicing has four main types: intron retention, exon skipping, alterative 5' splice site or alternative donor site, and alternative 3' splice site or alternative acceptor site. SplicingTypesAnno is an R package to annotate these four major splicing types by RNA-Seq data. As a post-processing tool after reads alignment, it annotates the alternative splicing events with details at the intron or exon level. Specially, It takes the alignment file, bam file as input, and analyzes the raw reads through the pipeline with searching algorithms, and defines the related alternative splicing types, finally generates a user-friendly web report for users. In addition, it provides high flexibility for users to handle large set of data by global-level and gene-level functions. In the global-level, users can make use of complex clusters with parallel computing feature to speed up the multiple sample analysis. In the gene-level, users can conveniently extract the related alternative splicing events with a single laptop. The output is stored as bed format, and easily to be visualized with IGV.

2 Installation

Please install:

1. Install packages from CRAN
```
install.packages(c("hwriter", "SortableHTMLTables"))
```

2. Install packages from Bioconductor
```
source("http://bioconductor.org/biocLite.R"); biocLite("Rsamtools")
```

3 Major splicing types

Alternative splicing includes exon skipping, intron retention, alternative donor, alternative acceptor, alternative both sites. There are also some other forms with small percentage, which is not discussed in this software. We design the searching algorithm for splicing types fully based on the structural properties of junction and non-junction reads (Figure 1). As a result, we use the known splicing sites as reference set. If the splicing sites derived from junction reads match the reference set, we consider them as "known sites"; otherwise as "novel sites". If the splicing sites are known, but the link between two sites is not reported from transcripts, this link is marked as "novel splicing link".

To simplify the analysis, we further divide these alternative splicing types to two different subtypes: type I and type II by comparing read information with the annotation files. Generally type I only consists of one intron or exon structure; type II consists of more than one intron or exon structures. More specifically, type I of exon skipping describes the alternative splicing events that only one exon is skipped; type II of exon skipping describes the events with multiple exons skipped (Figure 2). Type I of intron retention defines the events that only one intron is retained; type II defines those that more than one intron is retained (Figure 3). Type I of the alternative donor or acceptor covers those

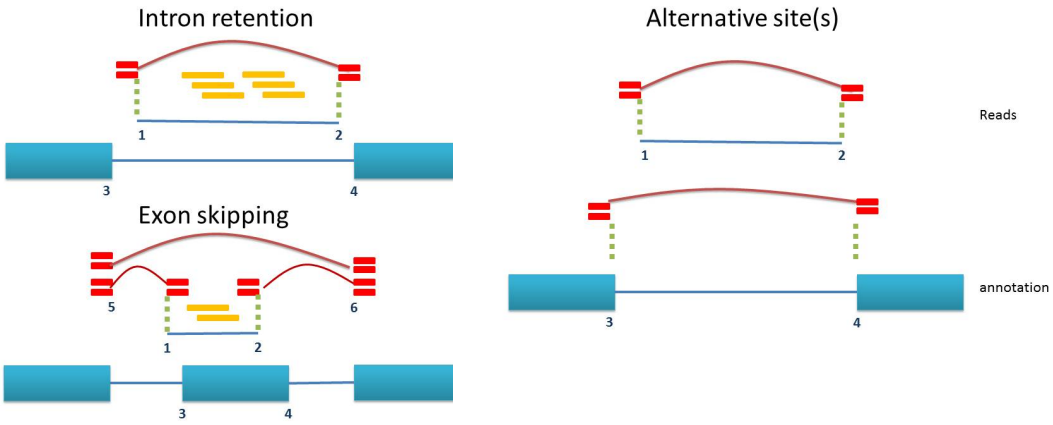

Figure 1: Major splicing types inferred from reads versus annotation from gtf/gff. Nomenclature of splicing sites used in the study - 1: novel left exon boundary; 2: novel right exon boundary; 3, 5: known left exon boundary; 4, 6: known right exon boundary.

events across only one intron; while type II defines other cases (Figure 4).

We use the following abbreviations:

- ri.1: type I of the intron retention.
- ri.2: type II of the intron retention.
- es.1: type I of the exon skipping.
- es.2: type II of the exon skipping.
- adleft.1: type I of the alternative donor site.
- adleft.2: type II of the alternative donor site.
- adright.1: type I of the alternative acceptor site.
- adright.2: type II of the alternative acceptor site.
- adboth.1: type I of the alternative both sites.
- adboth.2: type II of the alternative both sites.

4 Gene-level analysis and global-level analysis

4.1 Gene-level analysis

This section describes how to analyze data for one gene or a few genes. The package requires a gtf/gff file and an alignment bam file as input and can extract the sequencing data for this gene or a few genes from one bam files (one sample)

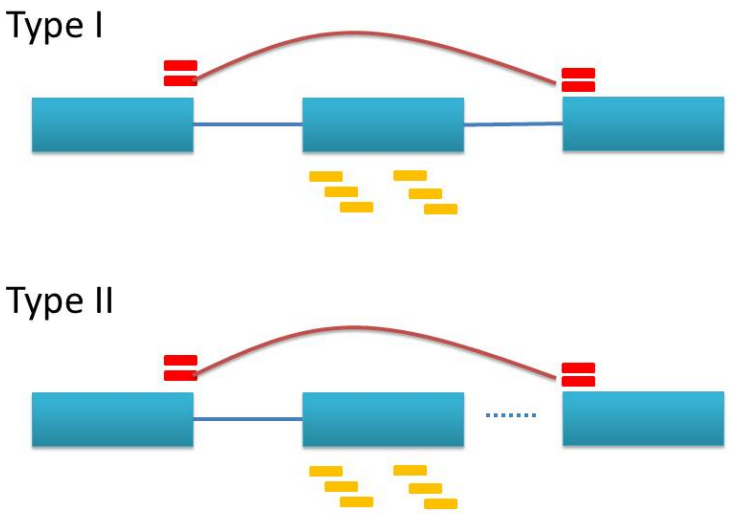

Figure 2: Two types of exon skipping: es.1 and es.2.

or multiple bam files (multiple samples). The final output can either be a list object with major splicing types or a user-friendly web report.

The analysis at this level includes two steps: 1) creating exon_intron structure by `translateGTF` function; 2) extracting related reads from bam file and annotating splicing types.

The main functions are `splicingGene`, `splicingReport`, `splicingCount`, `combineGene`, and `combineCount`. See details in the **Function description** section.

4.2 Global-level analysis

The analysis at this level focuses on how to globally analyze all genes for one sample or multiple samples. The package takes a gtf/gff file and an alignment bam file as input, and generates a convenient web report as output in the current working directory.

5 Input and output

5.1 Annotation file: gtf/gff file

For requirement of general format, see details at UCSC (http://genome.ucsc.edu/FAQ/FAQformat). For examples, use **?translateGTF** to see details.

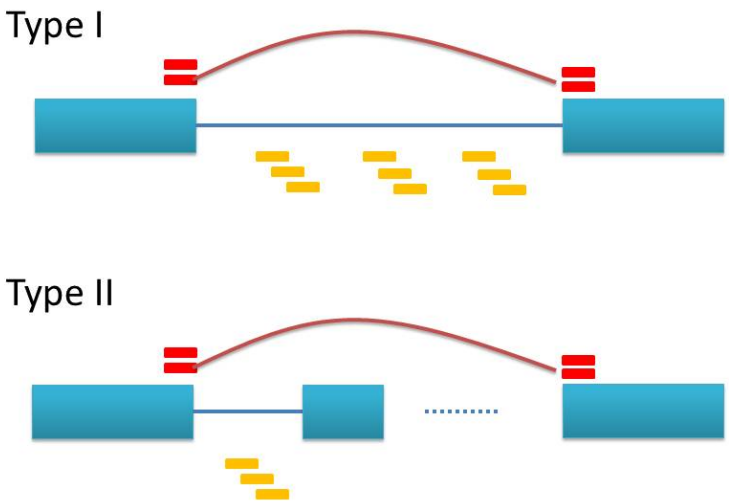

Figure 3: Two types of intron retention: ri.1 and ri.2.

The annotation file should NOT have header, and start as the first line. The file should be tab-delimited text file.

5.2 Alignment file: bam file

Each bam file represents one sample, generated from alignment software, such as bowtie or tophat. Paired-end data are treated as single-end data.

5.3 Output

The output columns are,

- **mergeID**: the identifier for merging this splicing type across multiple samples. The mergeID consists of chromosome number, genomic coordinates and strand information. The genomic coordinates are the two internal boundaries of the junction reads. For intron retention, the mergeID should be for an inferred intron, which is the retained intron for exon skipping, the mergeID should be for an inferred exon, which is the skipped exon; for alternative sites, the mergeID should be the gap of between two splicing junctions. It may or may not match the known annotation.

  e.g., chr2:24907270-24907551_+: 24907270 and 24907551 are site 1 and site 2 in Figure 1.

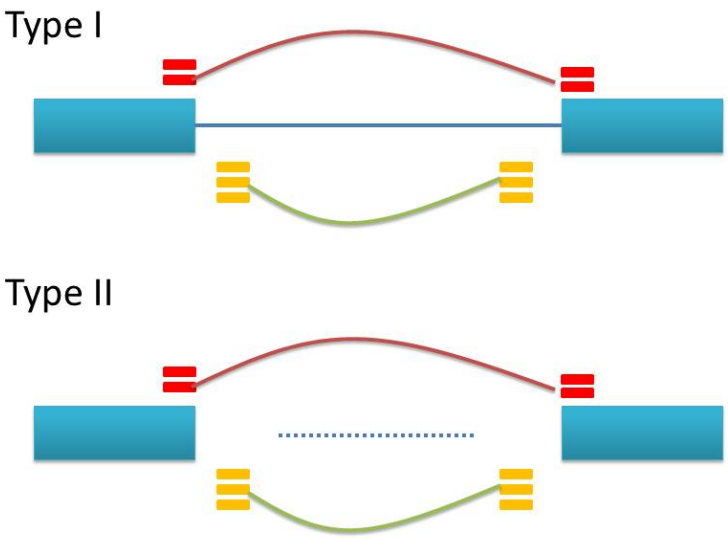

Figure 4: Two types of alternative both sites: adboth.1 and adboth.2.

- **geneName**: the gene name from gtf/gff file. Specifically, for gtf file, it should be "Attributes".

  e.g., gene_id Pnpla7_chr2_+_intron_30: for intron retention.
  e.g., gene_id LOC683722; gene_name LOC683722; p_id P10335; transcript_id NM_001101008; tss_id TSS1868;Exon_7: for exon skipping.

- **nonjun**: the number of non-junction reads invovled in this splicing type.
- **jun**: the number of junction reads invovled in this splicing type.
- **nonjun.norm**: the scaled number of non-junction reads, $(no*10^6)/(totalReadsOfBamFile* eventLength)$.
- **jun.norm**: the scaled number of junction reads, $(no*10^6)/(totalReadsOfBamFile* readLength*2)$.
- **ratio**: the percentage of splicing types. For intron retention, it should be $nonjun/(jun+nonjun)$; for exon skipping, it should be $jun/(nonjun+jun)$; for alternative sites, it should be $altersite1/(altersite1+altersite2)$.
- **junLeftBorder**: the end coordinate of the left junction. It is site 1 in Figure 1.
- **junRightBorder**: the start coordinate of the right junction. It is site 2 in Figure 1.

- **intronStart**: the start coordinate of the intron (for intron retention). It is site 3 in intron retention of Figure 1.
- **intronEnd**: the end cooridnate of the intron (for intron retention). It is site 4 in intron retention of Figure 1.
- **exonStart**: the start coordinate of the exon (for exon skipping). It is site 3 in exon skipping of Figure 1.
- **exonEnd**: the end coordinate of the exon (for exon skipping). It is site 4 in exon skipping of Figure 1.
- **count, countMax, countSum**: there are three metrics for measuring alternative acceptor/donor/both site(s): count, countSum, and countMax (for alternative sites). For one event of alternative site, it involves a few different sites inferred from a few different junction reads. To differentiate these, **count** is the number of reads for one specific site; **countMax** is the number of reads for one specific site which has the maximum number of junction reads; **countSum** is the number of total reads for all the sites.
- **junLeftEndCollection**: the coordinate and reads number collection for the 5' site of all the splicing types (for alternative site).

  e.g., 100031_100220,10_14: Let's assume that the splicing event is alternative donor site, and site 2 and site4 matches in Figure 1 . Then 100031 and 100220 are the site 1 and site 3 (5' site), and 10 and 14 are the read number for the junction read with site1, and the junction read with site 3 respectively.
- **junRightStartCollection**: the coordinate and reads number collection for the 3' site of all the splicing types (for alternative site).
- **novelSplicingLink**: the novel splicing links between two exons (for alternative site).
- **novelLeft**: the novel 5' site (for alternative site).
- **novelRight**: the novel 3' site (for alternative site).
- **note**: annotation for special case. For example, some intron retention may occur inside exon. This may happen when there are two samll exons overlapped with the third big exon.
- **note2**: summary sentence for ratio.

  e.g., 0.4167=100031_100220,10_14_SR813_adleft.1: **0.4167** is calculated from $10/(10 + 14)$; **100031, 100220** are two alternative sites for alternative donor sites; **10,14** are the read number for two alternative sites; **SR813** is the sample name for the bam file; **adleft.1** is the type I of the alternative donor site.

6 Function description

The main features of this package are 1) to translate gtf/gff file to exon_intron structure; 2) to count the reads for exon_intron structure; 3) to calculate and annotate the major splicing types. The first feature is implemented by `translateGTF`; the second feature requires two functions: `splicingCount` and `combineCount`. The last feature is achieved by two functions: `splicingGene` and `splicingReport`.

6.1 Translate gtf/gff file to exon

6.1.1 translateGTF

```
translateGTF(gtfFile,
             gtfChrFormat="chr", # "Chr", "Chr0", "chr0", "1"
             gtfGeneLabel="gene_id",
             gtfGeneValue= "g", # "g.1"
             gtfTranscriptLabel=NULL, # "Parent="  "transcript_id"
             exonString="exon",
             selectGenes=NULL,
             gtfColnames=c("chr", "source", "feature", "start",
                           "end","score", "strand", "frame",
                           "geneName"),
             geneOverlap="both",
             exonOverlap="yes"
             )
```

Translate gtf/gff file to the `GRanges` object with gene, exon and intron structure. It is a preprocessing tool for function `splicingGene`.

Use `? translateGTF` for details about parameters.

6.2 Count the reads for exon_intron structure

6.2.1 splicingCount

```
splicingCount(selectGene,
              bamFile = "accepted_hits.sort.bam",
              select.GRange,
              sampleName="sample",
              sampleID=1, type="any"
              )
```

Count the reads within gene, exon and intron for one gene. The selected GRange object, coming from `translateGTF`.

Use `? splicingCount` for details about parameters.

6.2.2 combineCount

```
combineCount(sample.list)
```

Combine the results from `splicingCount` for all samples.

Use `? combineCount` for details about parameters.

6.3 Calculate and annotate the major splicing types

6.3.1 splicingGene

```
splicingGene(selectGene,
                      bamFile,
                      select.GRange,
                      eventType=c("ri.1", "ri.2", "es.1", "es.2",
                                  "adleft.1", "adleft.2",
                                  "adright.1", "adright.2",
                                  "adboth.1", "adboth.2"),
                      sampleName="sample",
                      sampleID=1,
                      reportSummary=TRUE,
                      minReadCounts = 10,
                      ratioNorm=FALSE,
                      novel="both"
                      )
```

This function belongs to gene-level analysis. It only works on single gene, and helps users to process mutiple bam files in a few seconds. It is a convenient tool for those who do not have access to cluster computing environment.

6.3.2 splicingReport

```
splicingReport(inputData=sampleList,
                        gtfFile,
                        gtfChrFormat="chr",
                        gtfGeneLabel="gene_id",
                        gtfGeneValue="g",
                        gtfTranscriptLabel=NULL,
                        outputDir="html",
                        eventType=c("ri.1", "ri.2", "es.1", "es.2",
                                    "adleft.1", "adleft.2",
                                    "adright.1", "adright.2",
                                    "adboth.1", "adboth.2"),
                         inputFile=sampleFile,
                         minReadCounts=10,
                         selectGenes=NULL,
                         ratioNorm=FALSE,
                         novel="both",
                         parallel=FALSE, cpus=2
                        )
```

This function can be either gene-level analysis or global-level anaysis dependent on the `selectGenes` parameter. A web report will be generated in the current working directory.

Hardware requirement: for gene-level analysis, there is no specific requirement; for global-level analysis, this function requires a large amount of memory. Users can take advantage of parallel computing to speed up the analysis.

7 Demonstration

To illustrate how to use this package, two data sets are used(SRR094623 and SRR094624). The analysis pipeline includes, 1) translating gtf/gff3 to gene, exon and intron structure; 2) counting reads at the gene-level; 2) quantifying and annotating at the gene-level; 3) quantifying and annotating at the global-level.

7.1 Data sets

The raw data are from NCBI SRA GSE26561. The SRR094623 is for control, and the SRR094624 is for knockout. After aligning the raw data with tophat, we only selected the reads mapping to one gene: *Pnpla3*.

7.2 Analysis pipeline

1. We first convert the gff3 file to gene/exon/intron structure.

```
> options(width = 50)
> library(SplicingTypesAnno)
> mm9.pnpla7.gtfFile <- system.file("extdata",
+     "mm9_Pnpla7.gtf", package = "SplicingTypesAnno")
> result.GRange <- translateGTF(mm9.pnpla7.gtfFile,
+     gtfTranscriptLabel = "transcript_id")
```

```
[1] "Done..."
```

2. Then we count the reads for exon and intron respectively

```
> bam.1 <- system.file("extdata", "liver_ctr.sort.bam",
+     package = "SplicingTypesAnno")
> bam.2 <- system.file("extdata", "liver_ko.sort.bam",
+     package = "SplicingTypesAnno")
> selectGene <- "Pnpla7"
> sample1.c <- splicingCount(selectGene,
+     bam.1, result.GRange, sampleName = "liver_ctr",
+     sampleID = 1)
```

```
[1] "GENE - Pnpla7 are processed..."
[1] "some chrNo in bam file are NOT in gtf/gff3 file! The unmatched data will be overlooke
```

```
> sample2.c <- splicingCount(selectGene,
+     bam.2, result.GRange, sampleName = "liver_ko",
+     sampleID = 2)
```

```
[1] "GENE - Pnpla7 are processed..."
[1] "some chrNo in bam file are NOT in gtf/gff3 file! The unmatched data will be overlooke
```

3. To compare different samples, we combine all the results.

```
> sList.c <- list(sample1.c, sample2.c)
> ccount <- combineCount(sList.c)
```

4. In addition, we can quantify and annotate the major splicing types at the gene-level.

```
selectGene <- c("Pnpla7")
sample1 <- splicingGene(selectGene, bam.1, result.GRange,
                        sampleName="liver_ctr", sampleID=1
                        )

sample2 <- splicingGene(selectGene, bamFile2, result.GRange,
                        sampleName="liver_KO", sampleID=2
                        )

sList <- list(sample1$type, sample2$type)
ri.1 <- combineGene(sList, splicingEvents="ri.1", selectGenes)
ri.2 <- combineGene(sList, splicingEvents="ri.2", selectGenes)
es.1 <- combineGene(sList, splicingEvents="es.1", selectGenes)
es.2 <- combineGene(sList, splicingEvents="es.2", selectGenes)
adleft.1 <- combineGene(sList, splicingEvents="adleft.1", selectGenes)
adleft.2 <- combineGene(sList, splicingEvents="adleft.2", selectGenes)
adright.1 <- combineGene(sList, splicingEvents="adright.1", selectGenes)
adright.2 <- combineGene(sList, splicingEvents="adright.2", selectGenes)
adboth.1 <- combineGene(sList, splicingEvents="adboth.1", selectGenes)
adboth.2 <- combineGene(sList, splicingEvents="adboth.2", selectGenes)
```

5. Finally we can generate a user-friendly web report.

```
sampleList <- list(SampleName=c("liver_ctr", "liver_ko"),
                   BamFiles=c(bam.1, bam.2),
                   SampleID=c(1,2)
                    )
splicingReport(inputData=sampleList,
               gtfFile=mm9.pnpla7.gtfFile,
               selectGenes=c("Pnpla7"))
```

6. The bed files in the report can be visualized in IGV.