SPLINTER

Diana Low

15 April 2025

Package

SPLINTER 1.35.0

Contents

1 Introduction

SPLINTER provides tools to analyze alternative splicing sites, interpret outcomes based on sequence information, select and design primers for site validiation and give visual representation of the event to guide downstream experiments.

2 Loading the package

To load the *SPLINTER* package:

```
library(SPLINTER)
```

3 Initializing the genome for transcript selection

In this example, we will be utilizing the mm9 genome for mouse. You will need to install the appropriate package (eg. *BSgenome.Mmusculus.UCSC.mm9*) for the genome that you will be using.

```
library(BSgenome.Mmusculus.UCSC.mm9)
library(GenomicFeatures)
bsgenome <- BSgenome.Mmusculus.UCSC.mm9
```

We begin with full set of available transcripts to screen from, and read it into a *TxDb* object. One source of this (best option to ensure compatibility) would be the GTF file that you have used for alternative splicing analysis. For other sources of data, please refer to *GenomicFeatures*).

We then extract the coding sequences (CDS), and transcripts in general (coding and non-coding) (exons) from this object.

```
data_path<-system.file("extdata",package="SPLINTER")
gtf_file<-paste(data_path,"/Mus_musculus.Ensembl.NCBIM37.65.partial.gtf",sep="")

library(txdbmaker)
txdb <- makeTxDbFromGFF(file=gtf_file,chrominfo = Seqinfo(genome="mm9"))
## Warning in .makeTxDb_normarg_chrominfo(chrominfo): genome version information
## is not available for this TxDb object

# txdb generation can take quite long, you can save the object and load it the next time
# saveDb(txdb,file="txdb_hg19.sqlite")
# txdb<-loadDb(file="txdb_hg19.sqlite")

# extract CDS and exon information from TxDb object
thecds<-cdsBy(txdb,by="tx",use.names=TRUE)
theexons<-exonsBy(txdb,by="tx",use.names=TRUE)
```

4 Reading in the splicing analysis file

The output file from MATS is used here, but essentially all that is needed are coordinates of the exons (target and flanking) involved in the splicing processt to be studied. For the case of exon skipping, this will include the upstream, target and downstream exons. More output types will be supported in the future.

The following types of alternative splicing events are accepted:

Type of alternative splicing event	Definition
SE	Skipped exon
RI	Retained intron
MXE	Mutually exclusive exon
A5SS	Alternative 5' splice site
A3SS	Alternative 3' splice site

```
typeofAS="SE"
mats_file<-paste(data_path,"/skipped_exons.txt",sep="")
splice_data <-extractSpliceEvents(data=mats_file, filetype='mats', splicetype=typeofAS)
splice_data$data[,c(1:10)]
##                       ID             Symbol   chr strand exonStart  exonEnd
## 4825  ENSMUSG00000052337 ENSMUSG00000052337  chr6      +  71816720 71816734
## 17227 ENSMUSG00000023110 ENSMUSG00000023110 chr14      -  55132128 55132291
## 22583 ENSMUSG00000079477 ENSMUSG00000079477  chr6      -  87965626 87965712
## 13693 ENSMUSG00000024911 ENSMUSG00000024911 chr19      +   5464334  5464431
## 18826 ENSMUSG00000027940 ENSMUSG00000027940  chr3      +  89894935 89895013
##       upstreamStart upstreamEnd downstreamStart downstreamEnd
## 4825       71813135    71813264        71818574      71818797
## 17227      55130830    55130991        55133434      55133483
## 22583      87963632    87963692        87995067      87995239
## 13693       5464129     5464215         5464925       5465051
## 18826      89893932    89894001        89903450      89904487
```

4.1 Additional annotation

SPLINTER assumes that the main identifier is ENSEMBL, however gene symbols can be added.

```
splice_data<-addEnsemblAnnotation(data=splice_data,species="mmusculus")

# (Optional) Sorting the dataframe, if you have supporting statistical information
splice_data$data<-splice_data$data[with(splice_data$data,order(FDR,-IncLevelDifference)),]
head(splice_data$data[,c(1:10)])
##                   ID Symbol   chr strand exonStart  exonEnd upstreamStart
## 1 ENSMUSG00000052337   Immt  chr6      +  71816720 71816734      71813135
## 2 ENSMUSG00000023110  Prmt5 chr14      -  55132128 55132291      55130830
## 3 ENSMUSG00000079477   Rab7  chr6      -  87965626 87965712      87963632
## 4 ENSMUSG00000024911   Fibp chr19      +   5464334  5464431       5464129
## 5 ENSMUSG00000027940   Tpm3  chr3      +  89894935 89895013      89893932
##   upstreamEnd downstreamStart downstreamEnd
## 1    71813264        71818574      71818797
## 2    55130991        55133434      55133483
## 3    87963692        87995067      87995239
## 4     5464215         5464925       5465051
## 5    89894001        89903450      89904487
```

5 Analyzing a specific gene

5.1 Inspecting a single gene in more detail (single record)

Once we have defined the events, we will pick 1 event to analyze.

```
single_id='Prmt5'
pp<-which(grepl(single_id,splice_data$data$Symbol)) # Prmt5 has 1 record

splice_data$data[pp,c(1:6)] # show all records
##                    ID Symbol   chr strand exonStart  exonEnd
## 2 ENSMUSG00000023110  Prmt5 chr14      -  55132128 55132291

single_record<-splice_data$data[pp[1],]
```

5.2 Finding relevant transcripts from the ENSEMBL database

To reduce search complexity, we define the valid transcripts and coding sequences with regards to our gene of interest. We find that Prmt5 has 7 transcripts, 2 of which are coding sequences.

```
valid_tx <- findTX(id=single_record$ID,tx=theexons,db=txdb)

valid_cds<- findTX(id=single_record$ID,tx=thecds,db=txdb)
```

5.3 Constructing the region of interest (ROI)

The `makeROI` function will create a list containing `GRanges` objects for the splicing event. This will help identify and construct relevant outputs later.

This list contains the following information:

- type: type of alternative splicing event
- name: name of gene
- roi: GRanges object of the exon
- flank: GRanges object of the flanking exons
- roi_range: GRanges list containing
  - GRanges object of Type 1
  - GRanges object of Type 2

Type of alternative splicing	Type 1 representation	Type 2 representation (annotated only)
SE	isoform with event exon included	isoform with the exon skipped
RI	isoform with normal exon boundaries	isoform with the intron retained
MXE	isoform defined 1st (leftmost) in input	isoform defined 2nd in input
A5SS	isoform with longer exon	isoform with shorter exon
A3SS	isoform with longer exon	isoform with shorter exon

```
roi <- makeROI(single_record,type="SE")
roi
## $type
## [1] "SE"
##
## $name
## [1] "ENSMUSG00000023110"
##
## $roi
## GRanges object with 1 range and 1 metadata column:
##       seqnames            ranges strand | exon_rank
##          <Rle>         <IRanges>  <Rle> | <integer>
##   [1]    chr14 55132128-55132291      - |         1
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
##
## $flank
## GRanges object with 2 ranges and 1 metadata column:
##       seqnames            ranges strand | exon_rank
##          <Rle>         <IRanges>  <Rle> | <integer>
##   [1]    chr14 55133434-55133483      - |         2
##   [2]    chr14 55130830-55130991      - |         1
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
##
## $roi_range
## GRangesList object of length 2:
## [[1]]
## GRanges object with 3 ranges and 0 metadata columns:
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]    chr14 55133434-55133483      -
##   [2]    chr14 55132128-55132291      -
##   [3]    chr14 55130830-55130991      -
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
##
## [[2]]
## GRanges object with 2 ranges and 0 metadata columns:
```

```
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]    chr14 55133434-55133483      -
##   [2]    chr14 55130830-55130991      -
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

5.4 Finding transcripts that contain the ROI

At this juncture, we look for transcripts are compatible with the ROI. Compatibility is defined as having the exact cassette (matching upstream, target, downstream) exons. In the case of intron retention, this would just be the 2 exons flanking the intron.

We notice here that Prmt5 only has 1 compatible transcript involved in the event ROI, out of 7 transcripts (or 2 coding transcripts). There are no Type 2 transcripts, which means there are no annotated transcripts of Prmt5 containing the alternative event.

```
compatible_tx<- findCompatibleEvents(valid_tx,roi=roi,verbose=TRUE)
## Checking Type 1.....
## ENSMUST00000023873
## Checking Type 2.....
## No transcripts found!
## 1 match(es) from original 7 transcripts.

compatible_cds<- findCompatibleEvents(valid_cds,valid_tx,roi=roi,verbose=TRUE)
## Checking Type 1.....
## ENSMUST00000023873
## Checking Type 2.....
## No transcripts found!
## 1 match(es) from original 2 transcripts.
```

6 Simulating alternatively spliced products

6.1 Simulating the outcome of exon skipping by removing an exonic region

```
region_minus_exon <-removeRegion(compatible_cds$hits[[1]],roi)
```

6.2 Simulating the outcome of intron retention by inserting an intronic region

```
# Not relevant for this Prmt5 skipped exon example
region_plus_exon <-insertRegion(region_minus_exon,roi)
```

6.3 Comparing sequences before and after removal/insertion of a region

```
event<-eventOutcomeCompare(seq1=compatible_cds$hits[[1]],seq2=region_minus_exon,
                    genome=bsgenome,direction=TRUE,fullseq=FALSE)
##
## ### ENSMUST00000023873 ###
## Nonsense mediated decay.
## -   early termination
## middle insertion GILKPK (247-252)
## middle deletion LEIGADLP (206-213)
## middle deletion EPIK (224-227)
## middle deletion KAAIL (227-231)
## multiple mismatch sites
## 3' end mismatch : ALEIGADLPSNHVIDRWLGEPIKAAILPTSIFLTNKKGFPVLSKVQQRLIFRLLKLEVQFIITGTNHHSEKEFCSYLQYLEYLSQNR
## length : 637 AA to 242 AA

event
## $alignment
## Global-Local PairwiseAlignmentsSingleSubject (1 of 1)
## pattern:     MAAMAVGGAGGSRVSSGRDLNCVPEIADTLGA...GVL----FLP-----PVLGILKPKSPSTQCL*
## subject: [1] MAAMAVGGAGGSRVSSGRDLNCVPEIADTLGA...AILPTSIFLTNKKGFPVL------SKVQQRLI
## score: 1085.5
##
## $eventtypes
## [1] "(NMD)"
```

7 Designing primers to inspect splicing regions

7.1 Getting the DNA of the region of interest

This function will return the DNA of the ROI, with exons separated by "[]" (Primer3 notation) and the junction marked by `jstart`.

```
aa<-getRegionDNA(roi,bsgenome)
aa
## $seq
## [1] "GTGGCATAACTTTCGGACTCTGTGTGACTATAGCAAGAGAATTGCAGTAG[]TTGGAAGTGCAGTTTATCATCACGGGAACCAACCACCACTCAGAGAAG
##
## $jstart
## [1] 51
```

7.2 Using Primer3 to design primers for alternative splicing identification

We have included a helper function to run Primer3 from within `R`. You will need to define the path to your Primer3 installation. Refer to `?callPrimer3` for more details.

```
primers<-callPrimer3(seq=aa$seq,sequence_target = aa$jstart,size_range='100-500')
```

```
primers[,c(1:4)]
##   i    PRIMER_LEFT_SEQUENCE PRIMER_RIGHT_SEQUENCE PRIMER_LEFT_TM
## 1 0   ACTTTCGGACTCTGTGTGACT  TCATAGGCATTGGGTGGAGG         58.967
## 2 1   TGGCATAACTTTCGGACTCTG  GGAGTGGGGACTGCAGATAG         58.027
## 3 2  GCATAACTTTCGGACTCTGTGT  CTCCTTCTCTGAGTGGTGGT         58.673
## 4 3  TCGGACTCTGTGTGACTATAGC  ATTGGGTGGAGGGCGATTTT         59.056
## 5 4 GGACTCTGTGTGACTATAGCAAG   GCTCATAGGCATTGGGTGG         58.322
```

Alternatively, primers can be entered manually with the appropriate headers.

```
primers <- data.frame(PRIMER_LEFT_SEQUENCE="ACTTTCGGACTCTGTGTGACT",
                      PRIMER_RIGHT_SEQUENCE="TCATAGGCATTGGGTGGAGG",
                      stringsAsFactors=FALSE)
```

7.3 Checking primers coverage

As a confirmation, we can run the primers against the ROI to give the genomic location of the primer coverage.

```
cp<-checkPrimer(primers[1,],bsgenome,roi)

cp
## $total_span
## GRanges object with 1 range and 0 metadata columns:
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]    chr14 55130876-55133475      *
```

```
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
##
## $primer_left_span
## GRanges object with 1 range and 0 metadata columns:
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]    chr14 55133455-55133475      *
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
##
## $primer_right_span
## GRanges object with 1 range and 0 metadata columns:
##       seqnames            ranges strand
##          <Rle>         <IRanges>  <Rle>
##   [1]    chr14 55130876-55130895      *
##   -------
##   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

7.4 Predicting PCR results using the primers

`getPCRsizes` will give you the length of the PCR product produced by the set of primers.

```
pcr_result1<-getPCRsizes(cp,theexons)
pcr_result1
##                   ID  bp
## 1 ENSMUST00000023873 322

tx_minus_exon <-removeRegion(compatible_tx$hits[[1]],roi)
pcr_result2<-getPCRsizes(cp,tx_minus_exon)
pcr_result2
##                   ID  bp
## 1 ENSMUST00000023873 158
```

7.4.1 Selecting sizes relevant to splicing event (subset of getPCRsizes)

While `getPCRsizes` will return all possible PCR products for a given set of annotation, `splitPCRhit` will return PCR product sizes that are relevant to the splicing event in question.

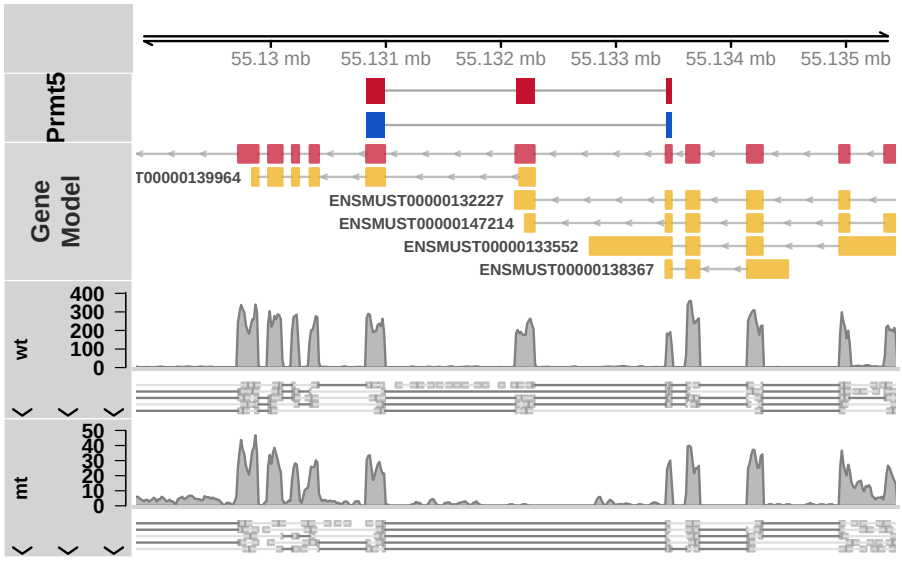
```
relevant_pcr_bands<-splitPCRhit(pcr_result1,compatible_tx)

relevant_pcr_bands
##                   ID  bp type
## 1 ENSMUST00000023873 322    1
```

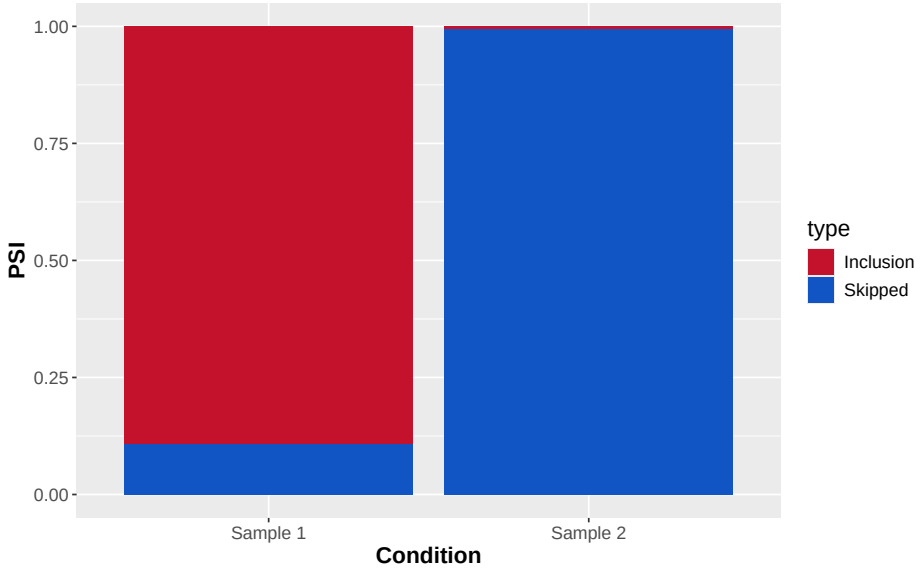
8 Plotting results

```
# reading in BAM files
mt<-paste(data_path,"/mt_chr14.bam",sep="")
wt<-paste(data_path,"/wt_chr14.bam",sep="")

# Plotting genomic range, read density and splice changes
eventPlot(transcripts=theexons,roi_plot=roi,bams=c(wt,mt),names=c('wt','mt'),
          annoLabel=single_id,rspan=2000)
```




```
# Barplot of PSI values if provided
psiPlot(single_record)
```



9 Session info

```
## R version 4.5.0 beta (2025-04-02 r88102)
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 24.04.2 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.22-bioc/R/lib/libRblas.so
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0  LAPACK version 3.12.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB              LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: America/New_York
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] txdbmaker_1.5.0                  GenomicFeatures_1.61.0
##  [3] AnnotationDbi_1.71.0             Biobase_2.69.0
##  [5] BSgenome.Mmusculus.UCSC.mm9_1.4.0 BSgenome_1.77.0
##  [7] rtracklayer_1.69.0               BiocIO_1.19.0
##  [9] Biostrings_2.77.0                XVector_0.49.0
## [11] GenomicRanges_1.61.0             GenomeInfoDb_1.45.0
## [13] IRanges_2.43.0                   S4Vectors_0.47.0
## [15] BiocGenerics_0.55.0              generics_0.1.3
## [17] SPLINTER_1.35.0                  BiocStyle_2.37.0
##
## loaded via a namespace (and not attached):
##   [1] RColorBrewer_1.1-3          rstudioapi_0.17.1
##   [3] jsonlite_2.0.0              magrittr_2.0.3
##   [5] farver_2.1.2                rmarkdown_2.29
##   [7] vctrs_0.6.5                 memoise_2.0.1
##   [9] Rsamtools_2.25.0            RCurl_1.98-1.17
##  [11] base64enc_0.1-3             tinytex_0.57
##  [13] htmltools_0.5.8.1           S4Arrays_1.9.0
##  [15] progress_1.2.3              curl_6.2.2
##  [17] SparseArray_1.9.0           Formula_1.2-5
##  [19] htmlwidgets_1.6.4           plyr_1.8.9
##  [21] Gviz_1.53.0                 httr2_1.1.2
##  [23] cachem_1.1.0                GenomicAlignments_1.45.0
##  [25] lifecycle_1.0.4             pkgconfig_2.0.3
##  [27] Matrix_1.7-3                R6_2.6.1
```

```
##  [29] fastmap_1.2.0               GenomeInfoDbData_1.2.14
##  [31] MatrixGenerics_1.21.0       digest_0.6.37
##  [33] colorspace_2.1-1            Hmisc_5.2-3
##  [35] RSQLite_2.3.9               seqLogo_1.75.0
##  [37] filelock_1.0.3              labeling_0.4.3
##  [39] httr_1.4.7                  abind_1.4-8
##  [41] compiler_4.5.0              bit64_4.6.0-1
##  [43] withr_3.0.2                 htmlTable_2.4.3
##  [45] backports_1.5.0             BiocParallel_1.43.0
##  [47] DBI_1.2.3                   biomaRt_2.65.0
##  [49] rappdirs_0.3.3              DelayedArray_0.35.0
##  [51] rjson_0.2.23                tools_4.5.0
##  [53] foreign_0.8-90              nnet_7.3-20
##  [55] glue_1.8.0                  restfulr_0.0.15
##  [57] grid_4.5.0                  checkmate_2.3.2
##  [59] cluster_2.1.8.1             gtable_0.3.6
##  [61] ensembldb_2.33.0            data.table_1.17.0
##  [63] hms_1.1.3                   xml2_1.3.8
##  [65] pillar_1.10.2               stringr_1.5.1
##  [67] dplyr_1.1.4                 BiocFileCache_2.17.0
##  [69] lattice_0.22-7              bit_4.6.0
##  [71] deldir_2.0-4                biovizBase_1.57.0
##  [73] tidyselect_1.2.1            googleVis_0.7.3
##  [75] knitr_1.50                  gridExtra_2.3
##  [77] bookdown_0.43               ProtGenerics_1.41.0
##  [79] SummarizedExperiment_1.39.0 xfun_0.52
##  [81] matrixStats_1.5.0           stringi_1.8.7
##  [83] UCSC.utils_1.5.0            lazyeval_0.2.2
##  [85] yaml_2.3.10                 evaluate_1.0.3
##  [87] codetools_0.2-20            interp_1.1-6
##  [89] tibble_3.2.1                BiocManager_1.30.25
##  [91] cli_3.6.4                   rpart_4.1.24
##  [93] munsell_0.5.1               dichromat_2.0-0.1
##  [95] Rcpp_1.0.14                 dbplyr_2.5.0
##  [97] png_0.1-8                   XML_3.99-0.18
##  [99] parallel_4.5.0              ggplot2_3.5.2
## [101] blob_1.2.4                  prettyunits_1.2.0
## [103] latticeExtra_0.6-30         jpeg_0.1-11
## [105] AnnotationFilter_1.33.0     bitops_1.0-9
## [107] pwalign_1.5.0               VariantAnnotation_1.55.0
## [109] scales_1.3.0                purrr_1.0.4
## [111] crayon_1.5.3                rlang_1.1.6
## [113] KEGGREST_1.49.0
```