

# Package ‘ASpli’

October 16, 2020

**Type** Package

**Title** Analysis of alternative splicing using RNA-Seq

**Version** 1.14.0

**Date** 2020-04-08

**Author**

Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky and Ariel Chernomoretz

**License** GPL

**biocViews** ImmunoOncology, GeneExpression, Transcription,  
AlternativeSplicing, Coverage, DifferentialExpression,  
DifferentialSplicing, TimeCourse, RNASeq, GenomeAnnotation,  
Sequencing, Alignment

**Depends** methods, grDevices, stats, utils, parallel, edgeR

**Imports** GenomicRanges, GenomicFeatures, BiocGenerics, IRanges,  
GenomicAlignments, Gviz, S4Vectors, AnnotationDbi, Rsamtools,  
BiocStyle

**Maintainer** Estefania Mancini <emancini@leloir.org.ar>

**Description** Integrative pipeline for the analysis of alternative  
splicing using RNAseq.

**git\_url** <https://git.bioconductor.org/packages/ASpli>

**git\_branch** RELEASE\_3\_11

**git\_last\_commit** 912b01b

**git\_last\_commit\_date** 2020-04-27

**Date/Publication** 2020-10-16

## R topics documented:

ASpli-package . . . . .	2
AS accessors . . . . .	4
ASDiscover . . . . .	5
ASpliAS-class . . . . .	6
ASpliCounts . . . . .	7
ASpliCounts-class . . . . .	8
ASpliDU-class . . . . .	9
ASpliFeatures-class . . . . .	9
binGenome . . . . .	10

binGenome-methods . . . . .	11
Counts accessors . . . . .	12
DU accessors . . . . .	13
DUreport . . . . .	14
DUreportBinSplice . . . . .	16
Examine ASpliDU objects . . . . .	18
Example data . . . . .	19
Features accessors . . . . .	20
filterDU . . . . .	21
getConditions . . . . .	23
junctionDUreport . . . . .	24
loadBAM . . . . .	26
mergeBinDUAS . . . . .	27
rds . . . . .	28
readCounts . . . . .	30
show-methods . . . . .	31
Subset ASpli objects . . . . .	31
write . . . . .	33
write-methods . . . . .	34

<b>Index</b>	<b>35</b>
--------------	-----------

---

ASpli-package

*Analysis of alternative splicing using RNAseq*


---

## Description

ASpli is an integrative and flexible package that facilitates the characterization of genome-wide changes in AS under different experimental conditions. ASpli analyzes the differential usage of introns, exons, and splice junctions using read counts, and estimates the magnitude of changes in AS by calculating differences in the percentage of exon inclusion or intron retention using splice junctions. This integrative approach allows the identification of changes in both annotated and novel AS events.

ASpli allows users to produce self-explanatory intermediate outputs, based on the aim of their analysis. A typical workflow involves parsing the genome annotation into new features called bins, overlapping read alignments against those bins, and inferring differential bin usage based on the number of reads aligning to the bins and junctions.

## Details

Package: ASpli

Type: Package

Version: 1.5.1

Date: 2018-02-22

License: GPL

Depends: methods, GenomicRanges, GenomicFeatures, edgeR, methods, BiocGenerics, IRanges, GenomicAlignments,

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky and Ariel Chernomoretz

**References**

- Acute effects of light on alternative splicing in light-grown plants. *Photochemistry and Photobiology*. Mancini, E, Sanchez, S, Romanowsky, A, Yanovsky, MJ. DOI: 10.1111/php.12550
- GEMIN2 attenuates the effects of temperature on alternative splicing and circadian rhythms in *Arabidopsis thaliana*. *Proceedings of the National Academy of Sciences*. Schlaen, RG, Mancini, E, Sanchez, SE, Perez-Santangelo, S, Rugnone, ML, Simpson, CG, Brown, JWS, Zhang, X, Chernomoretz, A, Yanovsky, MJ. DOI:10.1073/pnas.1504541112
- Genome wide comparative analysis of the effects of PRMT5 and PRMT4/CARM1 arginine methyltransferases on the *Arabidopsis thaliana* transcriptome. *BMC Genomics*. Hernando, E, Sanchez, S, Mancini, E, Yanovsky MJ. DOI:10.1186/s12864-015-1399-2
- A role for LSM genes in the regulation of circadian rhythms. *Proceedings of the National Academy of Sciences*. Perez Santangelo, S, Mancini, E, Francey, LJ, Schlaen, RG, Chernomoretz, A, Hogenesch, JB, Yanovsky MJ. DOI: 10.1073/pnas.1409791111
- The dengue virus NS5 protein intrudes in the cellular spliceosome and modulates splicing. *PLOS Pathogens*. De Maio, F, Risso, G, Iglesias, G, Shah, P, Pozzi, B, Gebhard, L, Mammi, L, Mancini, E, Yanovsky, M, Andino, R, Krogan, N, Srebrow, A and Gamarnik, A. DOI:10.1371/journal.ppat.1005841

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Calculate differential usage of genes, bins and junctions
du <- DUreport( counts, targets )
du <- junctionDUreport( counts, targets, appendTo = du )

# Calculate PSI / PIR for bins and junction.
```

```
as      <- AsDiscover( counts, targets, features, bams, readLength = 100,
                      threshold = 5, cores = 1 )
```

---

AS accessors

*Accessors for ASpliAS object*

---

## Description

Methods to retrieve and set data in ASpliAS object. Setting data into an ASpliAS object is not a typical task and must be done with care, because it can affect the integrity of the object.

## Usage

```
altPSI( x )
esPSI( x )
irPIR( x )
joint( x )
junctionsPIR( x )
junctionsPSI( x )
```

## Arguments

x                    An ASpliAS object

## Value

Returns dataframes with genomic metadata and PSI and PIR metrics

## Author(s)

Estefania Mancini, Marcelo Yanovsky, Ariel Chernomoretz

## Examples

```
# Accessing data tables from an ASpliAS object

as <- aspliASexample()

ap <- altPSI(as)
ep <- esPSI(as)
ip <- irPIR(as)
j  <- joint(as)
jpi <- junctionsPIR(as)
jps <- junctionsPSI(as)

# Setting data tables to an ASpliAS object

as2 <- new( 'ASpliAS' )

altPSI( as2 ) <- ap
esPSI( as2 ) <- ep
irPIR( as2 ) <- ip
joint( as2 ) <- j
```

```
junctionsPIR( as2 ) <- jpi
junctionsPSI( as2 ) <- jps
```

AsDiscover

*Report PSI and PIR using experimental junctions***Description**

Given a bin, it is possible to calculate PSI/PIR metric using junctions to estimate changes in the use of it along different conditions. PSI or PIR metric is calculated for each bin and experimental condition. The selection of which metric is used is based on the kind of splicing event associated with each bin.

**Usage**

```
AsDiscover( counts, targets, features, bam, readLength, threshold , cores )
```

**Arguments**

counts	An object of class ASpliCounts.
targets	A dataframe containing sample, bam and experimental factors as columns and samples as rows.
features	An object of class ASpliFeatures.
bam	A list with BAM files contents.
readLength	Read length of sequenced read. Default 100L
threshold	Minimum number of reads supporting junctions. Default=5
cores	Number of processing cores to use

**Value**

An object of class ASpliAS

irPIR	reports: event, eli counts (J1), ie1 counts (J2), j_within (J3), PIR by condition. J1, J2, J3 sum of junctions (J1, J2, J3) by condition.
altPSI	reports: event, J1 (start), J2 (end), J3 (exclusion), PSI. J1, J2, J3 sum of junctions (J1, J2, J3) by condition.
esPSI	reports: event, J1 (start), J2 (end), J3 (exclusion), PSI. J1, J2, J3 sum of junctions (J1, J2, J3) by condition.
junctionsPIR	PIR metric for each experimental junction using eli and ie2 counts. Exclusion junction is the junction itself. This output helps to discover new introns as well as new retention events.
junctionsPSI	Given a junction, it is possible to analyze if it shares start, end or both with another junction. If so, is because there is more than one way for/of splicing. Ratio between them along samples is reported.

**Author(s)**

Estefania Mancini, Marcelo Yanovsky and Ariel Chernomoretz

**See Also**

Accessors: [irPIR](#), [altPSI](#), [esPSI](#), [junctionsPIR](#), [junctionsPSI](#)

Export: [writeAS](#)

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Calculate differential usage of genes, bins and junctions
du <- DUreport( counts, targets )
du <- junctionDUreport( counts, targets, appendTo = du)

# Calculate PSI / PIR for bins and junction.
as <- AsDiscover( counts, targets, features, bams, readLength = 100,
                threshold = 5, cores = 1 )

writeAS( as = as, output.dir = "only_as" )
```

---

ASpliAS-class

*Class "ASpliAS"*


---

**Description**

Results of PSI and PIR using experimental junctions

**Slots**

**irPIR:** Reports: event, e1i counts (J1), ie1 counts (J2), j\_within (J3), PIR by condition. J1, J2, J3 sum of junctions (J1, J2, J3) by condition.

altPSI: Reports: event, J1 (start), J2 (end), J3 (exclusion), PSI. J1, J2, J3 sum of junctions (J1, J2, J3) by condition.

esPSI: Reports: event, J1 (start), J2 (end), J3 (exclusion), PSI. J1, J2, J3 sum of junctions (J1, J2, J3) by condition.

joint: It is a combination of irPIR, altPSI and esPSI tables

junctionsPIR: PIR metric for each experimental junction using e1i and ie2 counts. Exclusion junction is the junction itself. This output helps to discover new introns as well as new retention events

junctionsPSI: Given a junction, it is possible to analyze if it shares start, end or both with another junction. If so, is because there is more than one way for/of splicing. Ratio between them along samples is reported.

### Author(s)

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

### See Also

Methods: [ASDiscover](#)

Accessors: [altPSI](#), [irPIR](#), [esPSI](#), [joint](#), [junctionsPIR](#), [junctionsPSI](#)

---

ASpliCounts

*Class "ASpliCounts"*

---

### Description

Contains results of read overlaps against all feature levels summarization

### Slots

gene.counts

exon.intron.counts

junction.counts

e1i.counts

ie2.counts

gene.rd

bin.rd

### Author(s)

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

---

ASpliCounts-class      *Class "ASpliCounts"*

---

### Description

Contains results of read overlaps against all feature levels summarization

### Slots

gene.counts: Object of class "data.frame"  
exon.intron.counts: Object of class "data.frame"  
junction.counts: Object of class "data.frame"  
eli.counts: Object of class "data.frame"  
ie2.counts: Object of class "data.frame"  
gene.rd: Object of class "data.frame"  
bin.rd: Object of class "data.frame"

### Methods

**AsDiscover** psi and pir metrics  
**countsb** bin counts accesor  
**countseli** eli counts accesor  
**countsg** gene counts accesor  
**countsie2** ie2 counts accesor  
**countsj** junction counts accesor  
**DUreport\_DEXSeq** differential expression and usage estimation using DEXSeq  
**DUreport** differential expression and usage estimation using DEXSeq  
**rdsb** bin read densities accesor  
**rdsg** gen read densities accesor  
**rds** compute read densities on genes and bins  
**writeCounts** Export count tables  
**writeRds** Export read density tables

### Author(s)

Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz



---

ASpliDU-class	Class "ASpliDU"
---------------	-----------------

---

**Description**

Contains results of differential expression at gene level and differential usage at bin and junction level estimation using DEreport method.

**Slots**

genes

bins

junctions

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

---

ASpliFeatures-class	Class "ASpliFeatures"
---------------------	-----------------------

---

**Description**

Contains Genomic Ranges of different features extracted from a TxDb

**Slots**

genes:

bins:

junctions:

**Author(s)**

Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

binGenome

*Feature coordinates extraction***Description**

Exons and introns are subdivided into new features called exon and intron bins and are then classified into exclusively exonic bins, exclusively intronic bins or alternative splicing (AS) bins .

**Usage**

```
binGenome(genome, geneSymbols, logTo = "ASpli_binFeatures.log" )
```

**Arguments**

genome	An object of class transcriptDb (TxDb)
geneSymbols	A dataframe with symbol (common names) of TxDb genes. If geneSymbols is NULL, gene name will be repeated
logTo	Filename where to print features extraction log

**Details**

Exon and intron coordinates are extracted from gene annotation, only those from multi-exonic genes are saved for further evaluation. In case more than one isoform exist, some exons and introns will overlap. Exons and introns are then disjoint into new features called exon and intron bins, and then they are classified into exclusively exonic bins, exclusively intronic bin or alternative splicing bins (AS-bins), which are labeled according to which alternative splicing event are assumed to come from:

- ES: exon skipping
- IR: intron retention
- Alt5|3'ss: alternative five/three prime splicing site
- "\*" (ES\*, IR\*, AltSS\*) means this AS bin/region is involved simultaneously in more than one AS event type
- external: from the beginning or the end of a transcript

Subgenic features are labeled as follow (hypothetical GeneAAA):

- GeneAAA:E001: defines first exonic bin
- GeneAAA:I001: defines first intronic bin
- GeneAAA:Io001: defines first intron before disjoint into bins
- GeneAAA:J001: defines first junction

Junctions are defined as the last position of five prime exon (donor position) and first position of three prime exon (acceptor position). Using TxDb object, it is possible to extract annotated/known junctions. This information will be useful for the analysis of "experimental" junctions (reads aligned with gaps). Bins and junctions are labelled always in 5' to 3' sense. This notation is strand independent. It implies that bin / junction with lower numbering is always at 5'.

**Value**

An ASpliFeatures object. It is a list of features using GRanges format.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

[featuresg](#), [featuresb](#), [featuresj](#)

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Extract gene, bin and junctions features
GeneCoord <- featuresg(features)
BinCoord <- featuresb(features)
JunctionCoord <- featuresj(features)
```

---

binGenome-methods

*Feature coordinates extraction*

---

**Description**

Feature coordinates extraction from a Transcript Database

**Methods**

`signature(genome = "TxDb")` An object of class transcriptDb (TxDb)

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

[featuresg](#), [featuresb](#) , [featuresj](#)

**Description**

Accessors for ASpliCounts object

**Usage**

```
countsb(x)
countse1i(x)
countsg(x)
countsie2(x)
countsj(x)
rdsg(x)
rdsb(x)
```

**Arguments**

x                    An ASpliCounts object

**Value**

Returns dataframes with counts by sample and genomic metadata

**Author(s)**

Estefania Mancini, Marcelo Yanovsky, Ariel Chernomoretz

**Examples**

```
# Get data tables from an ASpliCounts object

counts <- aspliCountsExample()

cb1 <- countsb(counts)
ce1i <- countse1i(counts)
cg <- countsg(counts)
cie2 <- countsie2(counts)
cj <- countsj(counts)
rg <- rdsg(counts)
rb <- rdsb(counts)

# Set data tables to an ASpliCounts object

countsb(counts) <- cb1
countse1i(counts) <- ce1i
countsg(counts) <- cg
countsie2(counts) <- cie2
countsj(counts) <- cj
rdsg(counts) <- rg
rdsb(counts) <- rb
```

---

DU accessors	<i>Accessors for ASpliDU object</i>
--------------	-------------------------------------

---

**Description**

Accessors for ASpliDU object

**Usage**

```
genesDE( x )  
binsDU( x )  
junctionsDU( x )
```

**Arguments**

x                    An ASpliDU object

**Value**

Returns dataframes with genomic metadata and logFC and pvalue

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**Examples**

```
# Get data tables from an ASpliDU object  
  
du <- aspliDUexample1()  
  
gde <- genesDE( du )  
bdu <- binsDU( du )  
jdu <- junctionsDU( du )  
  
# Set data tables to an ASpliDU object  
  
genesDE( du )     <- gde  
binsDU( du )     <- bdu  
junctionsDU( du ) <- jdu
```

**Description**

Estimate differential expression at gene level and differential usage at bin level. When targets has only two conditions, and contrast is not set, the estimation of differential expression and usage is done with an exact test, otherwise is estimated using a generalized linear model.

**Usage**

```
DUreport( counts,
          targets,
          minGenReads = 10,
          minBinReads = 5,
          minRds = 0.05,
          offset = FALSE,
          offsetAggregateMode = c( "geneMode", "binMode" )[1],
          offsetUseFitGeneX = TRUE,
          contrast = NULL,
          forceGLM = FALSE,
          ignoreExternal = TRUE,
          ignoreIo = TRUE,
          ignoreI = FALSE,
          filterWithContrasted = FALSE,
          verbose = FALSE)
```

**Arguments**

counts	An object of class ASpliCounts
targets	A data.frame containing sample, bam and experimental factor columns.
minGenReads	Genes with at least an average of minGenReads reads for any condition are included into the differential expression test. Bins from genes with at least an average of minGenReads reads for all conditions are included into the differential bin usage test. Default value is 10 reads.
minBinReads	Bins with at least an average of minGenReads reads for any condition are included into the differential bin usage test. Default value is 5 reads.
minRds	Genes with at least an average of minRds read density for any condition are included into the differential expression test. Bins from genes with at least an average of minRds read density for all conditions are included into the differential bin usage test. Bins with at least an average of minRds read density for any condition are included into the differential bin usage test. Default value is 0.05.
ignoreExternal	Ignore Exon Bins at the beginning or end of the transcript. Default value is TRUE.
ignoreIo	Ignore original introns. Default TRUE
ignoreI	Ignore intron bins, test is performed only for exons. Default FALSE
offset	Corrects bin expression using an offset matrix derived from gene expression data. Default = FALSE

offsetAggregateMode	Choose the method to aggregate gene counts to create the offset matrix. When offsetAggregateMode is 'geneMode' and option offsetUseFitGeneX is TRUE, a generalized linear model is used to create the offset matrix. When offsetAggregateMode is 'geneMode' and option offsetUseFitGeneX is FALSE, the offset matrix is generated by adding a prior count to the gene count matrix. When offsetAggregateMode is 'binMode' a matrix from obtained from the sum of exonic bin counts, this only takes those bins that passes filters using minGenReads, minBinReads and minRds. Options:=c( "geneMode", "binMode" )[ 1 ] ,
offsetUseFitGeneX	Default= TRUE
contrast	Define the comparison between conditions to be tested. contrast should be a vector with length equal to the number of experimental conditions defined by targets. The values of this vector are the coefficients that will be used to weight each condition, the order of the values corresponds to the order given by getConditions function. When contrast is NULL, defaults to a vector containing -1, as the first value, 1 as the second and zero for all the remaining values, this corresponds to a pair comparison where the first condition is assumed to be a control and the second condition is the treatment, all other conditions are ignored. Default = NULL
forceGLM	Force the use of a generalized linear model to estimate differential expression and usage. Default = FALSE
filterWithContrasted	A logical value specifying if bins, genes and junction will be filtered by read quantity and read density using data from those conditions that will be used in the comparison, i.e. those which coefficients in contrast argument are different from zero. The default value is FALSE, it is strongly recommended to do not change this value.
verbose	A logical value that indicates that detailed information about each step in the analysis will be presented to the user.

**Value**

An ASpliDU object with results at genes, bins level.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

[edgeR](#), [junctionDUreport](#) Accessors: [genesDE](#), [binsDU](#) Export: [writeDU](#)

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata', 'genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )
```

```

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                     maxISize = 50000 )

# Calculate differential usage of genes and bins
du <- DUreport( counts, targets )

# Export results
writeDU( du = du, output.dir = "only_du" )

```

---

DUreportBinSplice

*Differential gene expression and differential bin usage estimation*


---

## Description

Estimate differential expression at gene level and differential usage at bin level using `diffSpliceDGE` function from `edgeR` package. This is an alternative approach to `DUreport`. The results at gene level are the same as the results from `DUreport`. The results at bin level are slightly different.

## Usage

```

DUreportBinSplice( counts,
                  targets,
                  minGenReads = 10,
                  minBinReads = 5,
                  minRds = 0.05,
                  contrast = NULL,
                  forceGLM = FALSE,
                  ignoreExternal = TRUE,
                  ignoreIo = TRUE,
                  ignoreI = FALSE,
                  filterWithContrasted = FALSE,
                  verbose = TRUE )

```

## Arguments

<code>counts</code>	An object of class <code>ASpliCounts</code>
<code>targets</code>	A dataframe containing sample, bam and experimental factor columns.



minGenReads	Genes with at least an average of minGenReads reads for any condition are included into the differential expression test. Bins from genes with at least an average of minGenReads reads for all conditions are included into the differential bin usage test. Default value is 10 reads.
minBinReads	Bins with at least an average of minGenReads reads for any condition are included into the differential bin usage test. Default value is 5 reads.
minRds	Genes with at least an average of minRds read density for any condition are included into the differential expression test. Bins from genes with at least an average of minRds read density for all conditions are included into the differential bin usage test. Bins with at least an average of minRds read density for any condition are included into the differential bin usage test. Default value is 0.05.
ignoreExternal	Ignore Exon Bins at the beginning or end of the transcript. Default value is TRUE.
ignoreIo	Ignore original introns. Default TRUE
ignoreI	Ignore intron bins, test is performed only for exons. Default FALSE
contrast	Define the comparison between conditions to be tested. contrast should be a vector with length equal to the number of experimental conditions defined by targets. The values of this vector are the coefficients that will be used to weight each condition, the order of the values corresponds to the order given by getConditions function. When contrast is NULL, defaults to a vector containing -1, as the first value, 1 as the second and zero for all the remaining values, this corresponds to a pair comparison where the first condition is assumed to be a control and the second condition is the treatment, all other conditions are ignored. Default = NULL
forceGLM	Force the use of a generalized linear model to estimate differential expression. It is not used to differential usage of bins. Default = FALSE
filterWithContrasted	A logical value specifying if bins, genes and junction will be filtered by read quantity and read density using data from those conditions that will be used in the comparison, i.e. those which coefficients in contrast argument are different from zero. The default value is FALSE, it is strongly recommended to do not change this value.
verbose	A logical value that indicates that detailed information about each step in the analysis will be presented to the user.

**Value**

An ASpliDU object with results at genes, bins level.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

[edgeR](#), [junctionDUreport](#) Accessors: [genesDE](#), [binsDU](#) Export: [writeDU](#)

**Examples**

```

# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Calculate differential usage of genes and bins
du <- DUreportBinSplice( counts, targets )

# Export results
writeDU( du = du, output.dir = "only_du" )

```

---

Examine ASpliDU objects

*Examine ASpliDU objects*

---

**Description**

ASpliDU object may contain results of differential expression of genes, differential usage of bins and junctions, however not everything is calculated at the same or even present. Calculations for genes and bins can be done independently from junctions. Functions `containsJunctions` and `containsGenesAndBins` allow to interrogate an ASpliDU object about the kind of results it contain.

**Usage**

```

containsJunctions( du )
containsGenesAndBins( du )

```

**Arguments**

`du`                    An ASpliDU object.

**Value**

A logical value that indicates that results for genes and bins, or results for junctions are available in the object.

**Author(s)**

Estefania Mancini, Marcelo Yanovsky, Ariel Chernomoretz

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Create an ASpliDU object.
#
du <- DUreport( counts, targets )

# Verify if du contains results for genes and bins.
containsGenesAndBins( du )
containsJunctions( du )

# Append junction results
du <- junctionDUreport( counts, targets, appendTo = du )

# Verify if du contains results for genes and bins.
containsJunctions( du )
```

**Description**

ASpli includes functions to easily build ASpli objects, used in examples in the vignette and man pages.

**Usage**

```
aspliASexample()  
aspliBamsExample()  
aspliCountsExample()  
aspliDUexample1()  
aspliDUexample2()  
aspliExampleBamList()  
aspliExampleGTF()  
aspliFeaturesExample()  
aspliJunctionDUexample()  
aspliTargetsExample()
```

**Value**

An ASpli object with example data.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**Examples**

```
as <- aspliASexample()  
bams <- aspliBamsExample()  
counts <- aspliCountsExample()  
du1 <- aspliDUexample1()  
du2 <- aspliDUexample2()  
bamfiles <- aspliExampleBamList()  
gtffile <- aspliExampleGTF()  
features <- aspliFeaturesExample()  
jdu <- aspliJunctionDUexample()  
targets <- aspliTargetsExample()
```

---

Features accesors

*Accessors for ASpliFeatures object*

---

**Description**

Accessors for ASpliFeatures object

**Usage**

```
featuresg( x )  
featuresb( x )  
featuresj( x )
```

**Arguments**

x                    An ASpliFeatures object

**Value**

Returns a GenomicRanges object. Function featuresg returns a GRangesList object containing exon ranges for each gene. Functions featuresb and featuresj, returns GRanges object for all bins and junctions.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**Examples**

```
# Get data from an ASpliFeatures object

features <- aspliFeaturesExample()

fg <- featuresg( features )
fb <- featuresb( features )
fj <- featuresj( features )

# Set data to an ASpliFeatures object

featuresg( features ) <- fg
featuresb( features ) <- fb
featuresj( features ) <- fj
```

---

filterDU

*Filtering ASpliDU objects*

---

**Description**

ASpliDU object can be filtered to retain genes, bins or junction according to their fdr corrected p-value estimated and log-fold-change.

**Usage**

```
filterDU(
  du ,
  what = c( 'genes', 'bins', 'junctions' ),
  fdr = 1,
  logFC = 0,
  absLogFC = TRUE,
  logFCgreater = TRUE
)
```

**Arguments**

du	An ASpliDU object
what	A character vector that specifies the kind of features that will be filtered. Accepted values are 'genes', 'bins', 'junctions'. Multiple values can be passed at the same time. The default value is c('genes','bins','junctions')
fdr	A double value representing the maximum accepted value of fdr corrected p-value to pass the filter. The default value is 1, the neutral value for fdr filtering operation.
logFC	A double value representing the cut-off for accepted values of log-fold-change to pass the filter. The default value is 0, the neutral value for logFC filtering operation if logFCgreater and absLogFC arguments are both TRUE.
absLogFC	A logical value that specifies that the absolute value of log-fold-change will be used in the filter operation. The default value is TRUE.
logFCgreater	A logical value that specifies that the log-fold-change value ( or abs(log-fold-change) if absLogFC argument is TRUE) of features must be greater than the cut-off value to pass the filter. The default value is TRUE.

**Value**

A new ASpliDU object with the results of the filtering operations. The elements of features that were not specified to be filtered are kept from the input ASpliDU object.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

[DUreport](#), [junctionDUreport](#), [DUreportBinSplice](#),

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )
```

```
# Read counts from bam files
counts  <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                      maxISize = 50000 )

# Calculate differential usage of junctions only
du      <- DUreport( counts, targets )

# Filter by FDR
duFiltered1 <- filterDU( du, what=c('genes','bins'),
                        fdr = 0.01 )

# Filter by logFC, only those that were up-regulated
duFiltered2 <- filterDU( du, what=c('genes','bins'),
                        logFC = log( 1.5, 2 ), absLogFC = FALSE )
```

---

getConditions	<i>Retrieve condition names from a targets data frame.</i>
---------------	--

---

### Description

Targets data frame contains experimental factors values for each sample. This function generates a simple name for each unique condition resulting from the combination of all experimental factors. The order of the conditions given by `getConditions` is the same that in contrast argument of `DUreport` function.

### Usage

```
getConditions( targets )
```

### Arguments

`targets`            A dataframe containing sample, bam and experimental factors columns

### Value

A character vector with the names of the conditions derived from experimental factors.

### Author(s)

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

### See Also

[DUreport](#),

## Examples

```
# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' )

# Load reads from bam files.
# Return value is c('C', 'D') in this example.
conditions <- getConditions(targets)

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam", "A_C_3.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam", "A_D_3.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:8)),
  bam = file.path( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','C','D','D','D','D' ),
  factor2 = c( 'E','E','F','F','E','E','F','F' )

# Load reads from bam files.
# Return value is c("C_E", "C_F", "D_E", "D_F") in this example.
conditions <- getConditions(targets)
```

---

junctionDUreport

*Differential junction usage estimation*

---

## Description

Estimate differential usage at junction level. When targets has only two conditions, and contrast is not set, the estimation of differential expression and usage is done with an exact test, otherwise is estimated using a generalized linear model.

## Usage

```
junctionDUreport( counts,
                  targets,
                  appendTo = NULL,
                  minGenReads = 10,
                  minRds = 0.05,
                  threshold = 5,
                  offset = FALSE,
                  offsetUseFitGeneX = TRUE,
                  contrast = NULL,
                  forceGLM = FALSE)
```





```

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C', 'C', 'C', 'D', 'D', 'D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Calculate differential usage of junctions only
du <- junctionDUreport( counts, targets )

# Calculate differential usage of genes, bins and junctions
du2 <- DUreport( counts, targets )
du2 <- junctionDUreport( counts, targets, appendTo = du2 )

# Export results
writeDU( du = du, output.dir = "only_du" )

```

---

loadBAM

*Load BAM files*


---

## Description

Load BAM files into R session using a targets specification.

## Usage

```
loadBAM(targets, cores)
```

## Arguments

targets	A data frame containing sample, bam and experimental factors columns
cores	Number of processors to use

## Value

A list of GAlignments. Each element of the list correspond to a samples BAM file.

## Author(s)

Estefania Mancini, Marcelo Yanovsky, Ariel Chernomoretz

## Examples

```
# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )

targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )
```

---

 mergeBinDUAS

*Differential usage of bins and PSI/PIR.*


---

## Description

This function merges the results of differential usage of bins, from an ASpliDU object, with PSI/PIR and junction information, from an ASpliAS object. Also, a delta PSI/PIR value is calculated from a contrast.

## Usage

```
mergeBinDUAS( du,
              as,
              targets,
              contrast = NULL )
```

## Arguments

du	An object of class ASpliDU
as	An object of class ASpliAS
targets	A data frame containing sample, bam files and experimental factor columns.
contrast	Define the comparison between conditions to be tested. <code>contrast</code> should be a vector with length equal to the number of experimental conditions defined by <code>targets</code> . The values of this vector are the coefficients that will be used to weight each condition, the order of the values corresponds to the order given by <code>getConditions</code> function. When <code>contrast</code> is <code>NULL</code> , defaults to a vector containing -1, as the first value, 1 as the second and zero for all the remaining values, this corresponds to a pair comparison where the first condition is assumed to be a control and the second condition is the treatment, all other conditions are ignored. The default value is <code>NULL</code> .

## Value

A data frame containing feature, event, locus, locus\_overlap, symbol, gene coordinates, start of bin, end of bin, bin length, log-Fold-Change value, p-value, fdr corrected p-value, J1 inclusion junction, J1 junction counts for each sample, J2 inclusion junction, J2 junction counts for each sample, J3 exclusion junction, J3 junction counts for each sample, PSI or PIR value for each bin, and delta PSI/PIR.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

[ASpliDU](#), [ASpliAS](#)

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam",
                  "B_C_0.bam", "B_C_1.bam", "B_C_2.bam",
                  "B_D_0.bam", "B_D_1.bam", "B_D_2.bam" )

targets <- data.frame(
  row.names = paste0('Sample_',c(1:12)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'A', 'A', 'A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'B' ),
  factor2 = c( 'C', 'C', 'C', 'D', 'D', 'D', 'C', 'C', 'C', 'D', 'D', 'D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Calculate differential usage of genes and bins
du <- DUreport( counts, targets , contrast = c(1,-1,-1,1))

# Calculate PSI / PIR for bins and junction.
as <- AsDiscover( counts, targets, features, bams, readLength = 100,
                threshold = 5, cores = 1 )

mas <- mergeBinDUAS( du, as, targets, contrast = c(1,-1,-1,1) )
```

**Description**

Read density of gene and bins is the quotient between the number of read counts and the length of the feature. The results are appended into an ASpliCounts object that must be given as argument. The explicit calculation of read densities is usually not required because is automatically performed by readCounts function.

**Usage**

```
rds( counts, targets )
```

**Arguments**

counts	An ASpliCounts object
targets	A data frame containing sample, bam and experimental factors columns

**Value**

An ASpliCounts object containing read densities of genes and bins.

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Calculates read densities
counts <- rds( counts, targets )
```

---

readCounts                      *Summarize read overlaps*

---

## Description

Summarize read overlaps against all feature levels

## Usage

```
readCounts( features, bam, targets, cores = 1, readLength,
            maxISize, minAnchor = 10)
```

## Arguments

features	An object of class ASpliFeatures. It is a list of GRanges at gene, bin and junction level
bam	List of GAlignments objects corresponding to bam files of samples.
targets	A dataframe containing sample, bam and experimental factors columns
readLength	Read length of sequenced library. It is used for compute E1I and IE2 read summarization
maxISize	maximum intron expected size. Junctions longer than this size will be discarded
cores	Number of cores to use. Default is 1
minAnchor	Minimum percentage of read that should be aligned to an exon-intron boundary

## Value

An object of class ASpliCounts. Each slot is a dataframe containing features metadata and read counts. Summarization is reported at gene, bin, junction and intron flanking regions (E1I, IE2).

## Author(s)

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

## See Also

Accessors: [countsg](#), [countsb](#), [countsj](#), [countseli](#), [countsie2](#), [rdsg](#), [rdsb](#), Export: [writeCounts](#)

## Examples

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
```

```
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
  maxISize = 50000 )

# Export data
writeCounts( counts, output.dir = "only_counts" )
```

---

show-methods

*Display a summary of data contained in ASpliObjects*

---

### **Description**

Display a summary of data contained in ASpliObjects

### **Details**

Display a summary of data contained in ASpliObjects

### **Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

---

Subset ASpli objects

*Subset ASpli objects*

---

### **Description**

ASpli provides utility functions to easy subset ASpliCounts objects, ASpliAS objects, targets data frame and lists GAlignments generated with loadBAM function. The subset can be done selecting some of the experimental conditions or samples names ( but not both ).

### **Usage**

```
subset( x, ... )
subsetBams( x, targets, select )
subsetTargets( targets, select, removeRedundantExpFactors )
```

**Arguments**

<code>x</code>	An ASpliCount or ASpliAS object for subset function, or list of GAlignments for subsetBams function.
<code>targets</code>	A dataframe containing sample, bam and experimental factor columns.
<code>select</code>	A character vector specifying the conditions or samples to be kept after subset operation. It's assumed that condition names are different from sample names.
<code>removeRedundantExpFactors</code>	When sub-setting the targets data frame, one or more experimental factors can have only one value. If this argument is TRUE those experimental factors are absent in the resulting target data frame.
<code>...</code>	Subsetting ASpliCounts and ASpliAS objects sub subset method requires a targets argument and a select argument with the same specifications that the arguments with the same name in subsetBams and subsetTargets functions

**Value**

A data frame similar to `x` ( or `targets` for `subsetTargets`) with only the containing only the selected elements.

**Author(s)**

Estefania Mancini, Marcelo Yanovsky, Ariel Chernomoretz

**Examples**

```
# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.
library(GenomicFeatures)
genomeTxDb <- makeTxDbFromGFF( system.file('extdata','genes.mini.gtf',
                                         package="ASpli") )

# Create an ASpliFeatures object from TxDb
features <- binGenome( genomeTxDb )

# Define bam files, sample names and experimental factors for targets.
bamFileNames <- c( "A_C_0.bam", "A_C_1.bam", "A_C_2.bam",
                  "A_D_0.bam", "A_D_1.bam", "A_D_2.bam" )
targets <- data.frame(
  row.names = paste0('Sample_',c(1:6)),
  bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
  factor1 = c( 'C','C','C','D','D','D' ) )

# Load reads from bam files
bams <- loadBAM( targets )

# Read counts from bam files
counts <- readCounts( features, bams, targets, cores = 1, readLength = 100,
                    maxISize = 50000 )

# Create ASpliAS object
as <- AsDiscover( counts, targets, features, bams, readLength = 100,
                threshold = 5, cores = 1 )
```



```
# Define selection
select <- c('Sample_1', 'Sample_2', 'Sample_4', 'Sample_5')

# Subset target
targets2 <- subsetTargets( targets, select )

# Subset bams
bams2 <- subsetBams( bams, targets, select )

# Subset ASpliCounts object
counts2 <- subset( counts, targets, select )

# Subset ASpliAS object
as2 <- subset( as, targets, select )
```

---

write

*Write results*

---

## Description

Export tab delimited files in structured output

## Usage

```
writeCounts(counts, output.dir="counts")
writeRds(counts, output.dir="rds")
writeDU(du, output.dir="du")
writeAS(as, output.dir="as")
writeAll(counts, du, as, output.dir="output")
```

## Arguments

counts	An ASpliCounts object
as	An ASpliAS object
du	An ASpliDU object
output.dir	Name of output folder (new or existing)

## Value

Tab delimited files are exported in a tidy manner into output folder

## Author(s)

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

## See Also

[AsDiscover](#), [binGenome](#), [DUreport](#)

---

write-methods

*Write results*

---

**Description**

Export tab delimited files in structured output

**Details**

Tab delimited files are exported in a tidy manner into output folder

**Author(s)**

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

[AsDiscover](#), [binGenome](#), [DUreport](#)

# Index

## \* alternative splicing, RNA-seq, junctions ASpli-package, 2

altPSI, 6, 7  
altPSI (AS accessors), 4  
altPSI, ASpliAS-method (ASpliAS-class), 6  
altPSI<- (AS accessors), 4  
altPSI<-, ASpliAS, data.frame-method  
(ASpliAS-class), 6  
AS accessors, 4  
ASDiscover, 5, 7, 33, 34  
ASDiscover, ASpliCounts-method  
(ASpliCounts-class), 8  
ASpli (ASpli-package), 2  
ASpli-package, 2  
ASpliAS, 28  
ASpliAS (ASpliAS-class), 6  
ASpliAS-class, 6  
aspliASexample (Example data), 19  
aspliBamsExample (Example data), 19  
ASpliCounts, 7  
ASpliCounts-class, 8  
aspliCountsExample (Example data), 19  
ASpliDU, 28  
ASpliDU (ASpliDU-class), 9  
ASpliDU-class, 9  
aspliDUexample1 (Example data), 19  
aspliDUexample2 (Example data), 19  
aspliExampleBamList (Example data), 19  
aspliExampleGTF (Example data), 19  
ASpliFeatures-class, 9  
aspliFeaturesExample (Example data), 19  
aspliJunctionDUexample (Example data),  
19  
aspliTargetsExample (Example data), 19  
  
binGenome, 10, 33, 34  
binGenome, TxDb-method  
(binGenome-methods), 11  
binGenome-methods, 11  
binsDU, 15, 17  
binsDU (DU accessors), 13  
binsDU, ASpliDU-method (ASpliDU-class), 9  
binsDU<- (DU accessors), 13

binsDU<-, ASpliDU-method  
(ASpliDU-class), 9  
  
containsGenesAndBins (Examine ASpliDU  
objects), 18  
containsGenesAndBins, ASpliDU-method  
(ASpliDU-class), 9  
containsJunctions (Examine ASpliDU  
objects), 18  
containsJunctions, ASpliDU-method  
(ASpliDU-class), 9  
Counts accesors, 12  
countsb, 30  
countsb (Counts accesors), 12  
countsb, ASpliCounts-method  
(ASpliCounts-class), 8  
countsb<- (Counts accesors), 12  
countsb<-, ASpliCounts, data.frame-method  
(ASpliCounts-class), 8  
countse1i, 30  
countse1i (Counts accesors), 12  
countse1i, ASpliCounts-method  
(ASpliCounts-class), 8  
countse1i<- (Counts accesors), 12  
countse1i<-, ASpliCounts, data.frame-method  
(ASpliCounts-class), 8  
countsg, 30  
countsg (Counts accesors), 12  
countsg, ASpliCounts-method  
(ASpliCounts-class), 8  
countsg<- (Counts accesors), 12  
countsg<-, ASpliCounts, data.frame-method  
(ASpliCounts-class), 8  
countsie2, 30  
countsie2 (Counts accesors), 12  
countsie2, ASpliCounts-method  
(ASpliCounts-class), 8  
countsie2<- (Counts accesors), 12  
countsie2<-, ASpliCounts, data.frame-method  
(ASpliCounts-class), 8  
countsj, 30  
countsj (Counts accesors), 12  
countsj, ASpliCounts-method  
(ASpliCounts-class), 8

- countsj<- (Counts accesors), 12
- countsj<- ,ASpliCounts, data.frame-method  
(ASpliCounts-class), 8
- DU accesors, 13
- DUreport, 14, 22, 23, 25, 33, 34
- DUreport, ASpliCounts-method  
(ASpliCounts-class), 8
- DUreportBinSplice, 16, 22
- DUreportBinSplice, ASpliCounts-method  
(ASpliCounts-class), 8
- edgeR, 15, 17, 25
- esPSI, 6, 7
- esPSI (AS accesors), 4
- esPSI, ASpliAS-method (ASpliAS-class), 6
- esPSI<- (AS accesors), 4
- esPSI<- ,ASpliAS, data.frame-method  
(ASpliAS-class), 6
- Examine ASpliDU objects, 18
- Example data, 19
- Features accesors, 20
- featuresb, 11
- featuresb (Features accesors), 20
- featuresb, ASpliFeatures-method  
(ASpliFeatures-class), 9
- featuresb<- (Features accesors), 20
- featuresb<- ,ASpliFeatures, GRanges-method  
(ASpliFeatures-class), 9
- featuresg, 11
- featuresg (Features accesors), 20
- featuresg, ASpliFeatures-method  
(ASpliFeatures-class), 9
- featuresg<- (Features accesors), 20
- featuresg<- ,ASpliFeatures, GRangesList-method  
(ASpliFeatures-class), 9
- featuresj, 11
- featuresj (Features accesors), 20
- featuresj, ASpliFeatures-method  
(ASpliFeatures-class), 9
- featuresj<- (Features accesors), 20
- featuresj<- ,ASpliFeatures, GRanges-method  
(ASpliFeatures-class), 9
- filterDU, 21
- filterDU, ASpliDU-method  
(ASpliDU-class), 9
- genesDE, 15, 17
- genesDE (DU accesors), 13
- genesDE, ASpliDU-method (ASpliDU-class),  
9
- genesDE<- (DU accesors), 13
- genesDE<- ,ASpliDU, data.frame-method  
(ASpliDU-class), 9
- getConditions, 23
- irPIR, 6, 7
- irPIR (AS accesors), 4
- irPIR, ASpliAS-method (ASpliAS-class), 6
- irPIR<- (AS accesors), 4
- irPIR<- ,ASpliAS, data.frame-method  
(ASpliAS-class), 6
- joint, 7
- joint (AS accesors), 4
- joint, ASpliAS-method (ASpliAS-class), 6
- joint<- (AS accesors), 4
- joint<- ,ASpliAS, data.frame-method  
(ASpliAS-class), 6
- junctionDUreport, 15, 17, 22, 24
- junctionDUreport, ASpliCounts-method  
(ASpliCounts-class), 8
- junctionsDU, 25
- junctionsDU (DU accesors), 13
- junctionsDU, ASpliDU-method  
(ASpliDU-class), 9
- junctionsDU<- (DU accesors), 13
- junctionsDU<- ,ASpliDU, data.frame-method  
(ASpliDU-class), 9
- junctionsPIR, 6, 7
- junctionsPIR (AS accesors), 4
- junctionsPIR, ASpliAS-method  
(ASpliAS-class), 6
- junctionsPIR<- (AS accesors), 4
- junctionsPIR<- ,ASpliAS, data.frame-method  
(ASpliAS-class), 6
- junctionsPSI, 6, 7
- junctionsPSI (AS accesors), 4
- junctionsPSI, ASpliAS-method  
(ASpliAS-class), 6
- junctionsPSI<- (AS accesors), 4
- junctionsPSI<- ,ASpliAS, data.frame-method  
(ASpliAS-class), 6
- loadBAM, 26
- mergeBinDUAS, 27
- mergeBinDUAS, ASpliDU, ASpliAS-method  
(ASpliDU-class), 9
- rds, 28
- rds, ASpliCounts-method  
(ASpliCounts-class), 8
- rdsb, 30
- rdsb (Counts accesors), 12

rdsb, ASpliCounts-method  
     (ASpliCounts-class), 8  
 rdsb<- (Counts accesors), 12  
 rdsb<-, ASpliCounts, data.frame-method  
     (ASpliCounts-class), 8  
 rdsg, 30  
 rdsg (Counts accesors), 12  
 rdsg, ASpliCounts-method  
     (ASpliCounts-class), 8  
 rdsg<- (Counts accesors), 12  
 rdsg<-, ASpliCounts, data.frame-method  
     (ASpliCounts-class), 8  
 readCounts, 30  
 readCounts, ASpliFeatures-method  
     (ASpliFeatures-class), 9  
  
 show, ASpliAS-method (show-methods), 31  
 show, ASpliCounts-method (show-methods),  
     31  
 show, ASpliDU-method (show-methods), 31  
 show, ASpliFeatures-method  
     (show-methods), 31  
 show-methods, 31  
 subset (Subset ASpli objects), 31  
 Subset ASpli objects, 31  
 subset, ASpliAS-method (ASpliAS-class), 6  
 subset, ASpliCounts-method  
     (ASpliCounts-class), 8  
 subsetBams (Subset ASpli objects), 31  
 subsetTargets (Subset ASpli objects), 31  
  
 write, 33  
 write-methods, 34  
 writeAll (write), 33  
 writeAll, ANY-method (write-methods), 34  
 writeAll, ASpliCounts-method  
     (ASpliCounts-class), 8  
 writeAS, 6  
 writeAS (write), 33  
 writeAS, ASpliAS-method (ASpliAS-class),  
     6  
 writeAS-methods (write-methods), 34  
 writeCounts, 30  
 writeCounts (write), 33  
 writeCounts, ASpliCounts-method  
     (ASpliCounts-class), 8  
 writeCounts-methods (write-methods), 34  
 writeDU, 15, 17, 25  
 writeDU (write), 33  
 writeDU, ASpliDU-method (ASpliDU-class),  
     9  
 writeDU-methods (write-methods), 34  
 writeRds (write), 33  
  
 writeRds, ASpliCounts-method  
     (ASpliCounts-class), 8  
 writeRds-methods (write-methods), 34