Package ‘ASpli’

January 5, 2024

Type  Package
Title  Analysis of Alternative Splicing Using RNA-Seq
Version  2.12.0
Date  2023-08-02
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, limma, AnnotationDbi

Imports  GenomicRanges, GenomicFeatures, BiocGenerics, IRanges, GenomicAlignments, Gviz, S4Vectors, Rsamtools, BiocStyle, igraph, htmltools, data.table, UpSetR, tidyr, DT, MASS, grid, graphics, pbmcapply

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

Suggests

Maintainer  Ariel Chernomoretz <algo107@gmail.com>

git_url  https://git.bioconductor.org/packages/ASpli

git_branch  RELEASE_3_18

git_last_commit  bf94d05

git_last_commit_date  2023-10-24

Repository  Bioconductor 3.18

Date/Publication  2024-01-05
R topics documented:

ASpli-package .............................................. 3
AS accessors ............................................... 4
ASpli-deprecated ........................................... 6
ASpliAS-class ............................................... 6
ASpliCounts ............................................... 7
ASpliCounts-class ......................................... 8
ASpliDU-class ............................................... 9
ASpliFeatures-class ....................................... 9
ASpliIntegratedSignals-class ......................... 10
ASpliJDU-class ........................................... 10
ASpliSplicingReport-class ......................... 11
binGenome .................................................. 11
binGenome-methods .................................... 13
Counts accessors ....................................... 13
DU accessors ............................................ 14
DUreport ................................................... 15
DUreport.norm .......................................... 17
DUreport.offset ......................................... 19
DUreportBinSplice ..................................... 21
Examine ASpliDU objects ............................... 23
Example data ........................................... 24
exportIntegratedSignals ............................... 25
exportSplicingReports .................................. 27
Features accessors ..................................... 28
filterDU ................................................... 29
filterSignals ............................................. 31
gbCounts .................................................. 32
gbDUreport ................................................ 35
getConditions ............................................ 38
integratedSignals accessors ....................... 39
integrateSignals .......................................... 40
jCounts .................................................... 42
JDU accessors ............................................ 46
jDUreport ................................................... 46
junctionDUreport ....................................... 50
loadBAM .................................................. 52
mergeBinDUAS ........................................... 53
plotBins .................................................. 54
plotGenomicRegions .................................... 58
rds ......................................................... 61
show-methods .......................................... 62
splicingReport .......................................... 63
splicingReport accessors ............................ 64
Subset ASpli objects .................................... 65
write ....................................................... 66
write-methods ........................................... 67
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, Gviz

Author(s)

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

References


Examples

```
library(GenomicFeatures)
gtfFileName <- aspliExampleGTF()
genomeTxDb <- makeTxDbFromGFF( gtfFileName )
features <- binGenome( genomeTxDb )
BAMFiles <- aspliExampleBamList()
targets <- data.frame(  
  row.names = paste0('Sample',c(1:12)),
  bam = BAMFiles,
  f1 = c("A","A","A","A","A","A",
        "B","B","B","B","B","B"),
  f2 = c("C","C","C","D","D","D",
        "C","C","C","D","D","D"),
  stringsAsFactors = FALSE)
getConditions(targets)
mBAMs <- data.frame(bam = sub("_[02]","",targets$bam[c(1,4,7,10)]),
                    condition= c("A_C","A_D","B_C","B_D"))

gbcounts <- gbCounts( features = features,  
                        targets = targets,  
                        minReadLength = 100, maxISize = 50000,  
                        libType="SE",  
                        strandMode=0)

asd <- jCounts(counts = gbcounts,  
                features = features,  
                minReadLength = 100,  
                libType="SE",  
                strandMode=0)

gb <- gbDUreport(counts=gbcounts,  
                 contrast = c(1, -1, -1, 1 ))
jdur <- jDUreport(asd,  
                 contrast = c(1, -1, -1, 1 ),  
                 mergedBams = mBAMs)
sr <- splicingReport(gb, jdur, counts =gbcounts )
is <- integrateSignals(sr,asd)
```

---

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 )
junctionsPJU( x )

Arguments

x An ASpliAS object

Value

Returns dataframes with genomic metadata and PSI and PIR metrics

Author(s)

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

Examples

# Accessing data tables from an ASpliAS object

#as <- aspliASexample()

#as <- aspliASexample()

#ap <- altPSI(as)
#ep <- esPSI(as)
#ip <- irPIR(as)
#j <- joint(as)
#jpi <- junctionsPIR(as)
#jps <- junctionsPJU(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
#junctionsPJU( as2 ) <- jps
### ASPliAS-deprecated

**Deprecated functions in package ‘ASpli’**

**Description**

These functions are provided for compatibility with older versions of ‘ASpli’ only, and will be defunct at the next release.

**Details**

The following functions are deprecated and will be made defunct; use the replacement indicated below:

- `loadBAM`: `gbCounts`
- `readCounts`: `gbCounts`
- `AsDiscover`: `jCounts, splicingReport, integrateSignals`
- `DUreport`: `gbDUreport, jDUreport`
- `DUreportBinSplice`: `gbDUreport`
- `junctionDUreport`: `jDUreport`
- `mergeBinDUAS`: `splicingReport, integrateSignals`
- `junctionsPSI`: `junctionsPJU`
- `plotGenomicRegions`: `exportSplicingReports, exportIntegratedSignals`

### 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.
- `join`: 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
juncti
junctionsPJU: Given a junction, it is possible to analyze if it shares start, end or both with another
junction. If so, it is because there is more than one way for/of splicing. Ratio between them
along samples is reported.
targets: DataFrame with targets.
.ASpliVersion: ASpili version when this object was created. It should not be modified by the user.

Author(s)
Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also
Methods: AsDiscover
Accessors: altPSI, irPIR, esPSI, joint, junctionsPIR, junctionsPJU

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
condition.order

Author(s)
Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz
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"
e1i.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"
condition.order: Object of class "character"
targets: Object of class "data.frame"
.ASPliVersion: ASPli version when this object was created. It should not be modified by the user.

Methods

AsDiscover  psi and pir metrics
countsb  bin counts accesor
countse1i  e1i 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 acceesor
rds  compute read densities on genes and bins
writeCounts  Export count tables
writeRds  Export read density tables

Author(s)

Estefania Mancini, Andres Rabinovich, Javier Isete, Marcelo Yanovsky, Ariel Chernomoretz
**Description**

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

**Slots**

- genes
- bins
- junctions
- contrast
- \( .ASpliVersion \): ASpli version when this object was created. It should not be modified by the user.

**Author(s)**

Estefania Mancini, Andres Rabinovich, 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:
- transcriptExons:

**Author(s)**

Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz
**ASpliIntegratedSignals-class**

*Class* "ASpliIntegratedSignals"

**Description**

Contains results of differential expression at junction level.

**Slots**

- signals
- filters

`.ASpliVersion: ASpli version when this object was created. It should not be modified by the user.`

**Author(s)**

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

---

**ASpliJDU-class**

*Class* "ASpliJDU"

**Description**

Contains results of differential expression at junction level.

**Slots**

- localec
- localej
- anchorc
- anchorj
- jir
- jes
- jalt
- contrast

`.ASpliVersion: ASpli version when this object was created. It should not be modified by the user.`

**Author(s)**

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

Contains results of differential expression at junction level.

Slots

- binbased
- localebased
- anchorbased
- contrast

.ASpliVersion: ASpli version when this object was created. It should not be modified by the user.

Author(s)

Andres Rabinovich, Estefania Mancini, 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 = NULL, logTo = "ASpli_binFeatures.log", cores = 1)
```

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
- cores: Number of cores to use in parallel when binning the genome
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 bins or alternative splicing bins (AS-bins), which are labeled according to which alternative splicing event are assumed to came 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 TxDDb 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.
library(GenomicFeatures)
gtfFileName <- aspliExampleGTF()
genomeTxDb <- makeTxDbFromGFF( gtfFileName )

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

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

---

### 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

---

### Counts accesors

Accessors for ASpliCounts object

### Description

Accessors for ASpliCounts object

### Usage

```r
counts(x)
countseli(x)
countsg(x)
countsie2(x)
countsj(x)
rdsg(x)
rdsb(x)
condition.order(x)
targets(x)
```
Arguments

x          An ASpliCounts object

Value

Returns dataframes with counts by sample and genomic metadata

Author(s)

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

Examples

```r
# Get data tables from an ASpliCounts object

counts <- aspliCountsExample()

#cb1 <- countsb(counts)
#ce1i <- countsei1(counts)
#cg <- countsg(counts)
#cie2 <- countsei2(counts)
#cj <- countsj(counts)
#rg <- rdsg(counts)
#rb <- rdsb(counts)
#co <- condition.order(counts)
#tg <- targets(counts)

# Set data tables to an ASpliCounts object

#countsb(counts) <- cb1
#countsei1(counts) <- ce1i
#countsg(counts) <- cg
#countsei2(counts) <- cie2
#countsj(counts) <- cj
#rdsg(counts) <- rg
#rdsb(counts) <- rb
```

---

**DU accessors**  
Accessors for ASpliDU object

**Description**

Accessors for ASpliDU object

**Usage**

```r
genesDE( x )
binsDU( x )
junctionsDU( x )
```
Arguments

- An ASpliDU object

Value

Returns dataframes with genomic metadata and logFC and pvalue

Author(s)

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

Examples

```r
# 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
```

---

**DUreport**

Differential gene expression and differential bin usage estimation

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

```r
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")
- **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 an 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

dgeR, junctionDUreport Accessors: genesDE, binsDU Export: writeDU

Examples

#This function has been deprecated and is no longer needed. Please see vignette for new pipeline.
DUreport.norm

Usage

DUreport.norm( counts,
    minGenReads = 10,
    minBinReads = 5,
    minRds = 0.05,
    contrast = NULL,
    ignoreExternal = TRUE,
    ignoreIo = TRUE,
    ignoreI = FALSE,
    filterWithContrasted = TRUE,
    verbose = FALSE,
    threshold = 5)

Arguments

counts An object of class ASpliCounts

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 an 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

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 TRUE, 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.

threshold  Default = 5

Value
An ASpliDU object with results at genes, bins level.

Author(s)
estefania Mancini, andres Rabinovich, javier iserte, marcelo yanovsky, Ariel Chernomoretz

See Also
edgeR, jDureport  Accessors: genesDE, binsDU  Export: writeDU

Examples

#check ASpli package examples

Usage

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

counts An object of class ASpliCounts

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")

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 an 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

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 TRUE, 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, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also

dgeR, jDURreport Accessors: genesDE, binsDU Export: writeDU

Examples

#check ASpli package example

```r
DUreportBinSplice
```

**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 DURreport. The results at gene level are the same as the results from DURreport. The results at bin level are slightly different.

**Usage**

```r
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

counts  An object of class ASpliCounts

targets  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 an 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
Examine ASpliDU objects

See Also

dgeR, junctionDUreport

Accessors: genesDE, binsDU

Export: writeDU

Examples

#This function has been deprecated. Please see vignette for new pipeline.

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

Examples

# see ASpli package
Example data

Example Aspli objects

Description

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

Usage

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

Value

An ASpli object with example data.

Author(s)

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

Examples

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

Description

Export integrated signals in an easy to analyze HTML table.

Usage

```r
exportIntegratedSignals( is, output.dir="is",
    sr, counts, features, asd,
    mergedBams,
    jCompletelyIncluded = FALSE, zoomRegion = 1.5,
    useLog = FALSE, tcex = 1, ntop = NULL,
    openInBrowser = FALSE,
    makeGraphs = TRUE, bforce=FALSE
)
```

Arguments

- **is**: An object of class ASpliIntegratedSignals
- **sr**: An object of class ASpliSplicingReport
- **counts**: An object of class ASpliCounts
- **features**: An object of class ASpliFeatures
- **asd**: An object of class ASpliAS
- **output.dir**: HTML reports output directory
- **mergedBams**: Dataframe with two columns, bams and conditions. Bams are paths to merged bams for each condition to be ploted.
- **jCompletelyIncluded**: If TRUE only plot junctions completely included in plot region. Else plot any overlapping junction in the region
- **zoomRegion**: Magnify plot region by this factor
- **useLog**: Plot counts log
- **tcex**: Text size
- **ntop**: Only show n top signals
- **openInBrowser**: Open reports in browser when done
- **makeGraphs**: Generate graphs in reports
- **bforce**: Force plot generation even if plot already exists

Value

Produces html reports
Author(s)
Andres Rabinovich, Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also


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
genos <- 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'))

# Read counts from bam files
gbcounds <- gbCounts( features = features,
  targets = targets, minReadLength = 100, maxISize = 50000,
  libType="SE", strandMode=0)
jcounts <- jCounts(counts = gbcounds, features = features,
  minReadLength = 100, libType="SE", strandMode=0)

# Test for factor1
gbPaired <- gbDUreport(gbcounds, contrast = c(1, -1))
jPaired <- jDUreport(jcounts, contrast = c(1, -1))

# Generate a splicing report merging bins and junctions DU
sr <- splicingReport(gbPaired, jPaired, gbcounds)
is <- integrateSignals(sr, jcounts)

# Make merged bams dataframe
mergedBamsFileName <- c("A_C.bam", "A_D.bam")
mergedBams <- data.frame(bams = system.file('extdata', mergedBamsFileName, package="ASpli" ),
  condition = c("C", "D"), stringsAsFactors = FALSE)
exportSplicingReports

# Export integrated signals
exportIntegratedSignals(is, output.dir = paste0(tempdir(), "/is"), sr, gbcounts,
features, jcounts, mergedBams, makeGraphs = TRUE, bforce = TRUE )

exportSplicingReports  Export splicing reports

Description
Export splicing reports in easy to analyze HTML tables.

Usage
exportSplicingReports( sr, output.dir="sr",
openInBrowser = FALSE, maxBinFDR = 0.2, maxJunctionFDR = 0.2 )

Arguments
 sr An object of class ASpliSplicingReport
 output.dir HTML reports output directory
 openInBrowser Open reports in browser when done
 maxBinFDR Only show bins with FDR < maxBinFDR
 maxJunctionFDR Only show junctions with FDR < maxJunctionFDR

Value
Produces html reports

Author(s)
Andres Rabinovich, Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also
gbDUreport, jDUreport, splicingReport, ASpliSplicingReport

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'),
  subject = c(0, 1, 2, 0, 1, 2))

# Read counts from bam files
gbcounts <- gbCounts(features = features,
  targets = targets,
  minReadLength = 100, maxISize = 50000,
  libType="SE",
  strandMode=0)
jcounts <- jCounts(counts = gbcounts,
  features = features,
  minReadLength = 100,
  libType="SE",
  strandMode=0)

# Test for factor1 controlling for paired subject
gbPaired <- gbDUreport(gbcounts, formula = formula(~subject+factor1))
jPaired <- jDUreport(jcounts, formula = formula(~subject+factor1))

# Generate a splicing report merging bins and junctions DU
sr <- splicingReport(gbPaired, jPaired, gbcounts)

# Export splicing report
exportSplicingReports(output.dir = paste0(tempdir(), "/sr"), sr)

---

**Features accesors**

Accessors for ASpliFeatures object

**Description**

Accessors for ASpliFeatures object

**Usage**

featuresg( x )
featuresb( x )
featuresj( x )
transcriptExons( 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, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

Examples

```r
# 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
```

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

```r
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, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also

DUreport.norm, DUreport.offset, jDUreport, gbDUreport,

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.norm( 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 )

---

### filterSignals

#### Filter signals

##### Description

Filter signals

##### Usage

filterSignals( sr,
               bin.FC = 3,
               bin.fdr = 0.05,
               nonunif = 1,
               bin.inclusion = 0.1,
               bjs.inclusion = 0.2,
               bjs.fdr = 0.1,
               a.inclusion = 0.3,
               a.fdr = 0.05,
               l.inclusion = 0.3,
               l.fdr = 0.05,
               DetectionSummary = FALSE)

##### Arguments

- **sr**: An object of class ASpliSplicingReport
- **bin.FC**: Description TODO
- **bin.fdr**: Description TODO
- **nonunif**: Description TODO
gbCounts

Value

TODO

Author(s)

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

See Also

ASpliSplicingReport

Examples

# Create a transcript DB from gff/gtf annotation file.
# Warnings in this examples can be ignored.

#as <- new("ASpliAS")
#targets <- 1
#a <- junctionDUreportExt(as, targets)

---

gbCounts Summarize read overlaps

Description

Summarize read overlaps against all feature levels

Usage

gbCounts( features, targets, minReadLength, 
maxISize, minAnchor = 10, libType="SE", 
strandMode=0, alignFastq = FALSE, dropBAM = FALSE)
**Arguments**

- **features**
  An object of class ASpliFeatures. It is a list of GRanges at gene, bin and junction level.

- **targets**
  A dataframe containing sample, bam and experimental factors columns.

- **minReadLength**
  Minimum read length of sequenced library. It is used for computing E1I and IE2 read summarization. Make sure this number is smaller than the maximum read length in every bam file, otherwise no E1I or IE2 will be found.

- **maxISize**
  Maximum intron expected size. Junctions longer than this size will be discarded.

- **minAnchor**
  Minimum percentage of read that should be aligned to an exon-intron boundary.

- **libType**
  Defines how reads will be treated according their sequencing library type (paired (PE, default) or single end (SE)).

- **strandMode**
  Controls the behavior of the strand getter. It indicates how the strand of a pair should be inferred from the strand of the first and last alignments in the pair. 0: strand of the pair is always *. 1: strand of the pair is strand of its first alignment. This mode should be used when the paired-end data was generated using one of the following stranded protocols: Directional Illumina (Ligation), Standard SOLiD. 2: strand of the pair is strand of its last alignment. This mode should be used when the paired-end data was generated using one of the following stranded protocols: dUTP, NSR, NNSR, Illumina stranded TruSeq PE protocol. For more information see ?strandMode.

- **alignFastq**
  Experimental (that means it’s highly recommended to leave the default, FALSE): executes an alignment step previous to Bam summarization. Useful if not enough space on local disks for beans so fasts can be aligned on the fly, even from a remote machine, and then the BAMs can be deleted after each summarization. If set to TRUE, targets data frame must have a column named alignerCall with complete call to aligner for each sample. IE: STAR --runMode alignReads -outSAMtype BAM SortedByCoordinate --readFilesCommand zcat --genomeDir /path/to/STAR/genome/folder -runThreadN 4 --outFileNamePrefix sample name --readFilesIn /path/to/R1 /path/to/R2. Output must match bam files provided in targets.

- **dropBAM**
  Experimental (that means it’s highly recommended to leave the default, FALSE): If alignFastq is TRUE, deletes BAMs after summarization. Used in conjunction with alignFastq to delete BAMs when there’s not enough free space on disk. Use with caution as it will delete all files in "bam" column in targets dataframe.

**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).

- **countsg**
  symbol: gene symbol
  locus_overlap: other genes overlapping this locus
  gene_coordinates: gene coordinates
  start: gene start
  end: gene end
  length: gene length
  effective_length: gene effective length
  From effective_length to the end, gene counts for all samples

- **countsb**
  feature: bin type
  event: type of event assigned by ASpli when bining
  locus: gene locus
  locus_overlap: genes overlapping the same locus
  symbol: gene symbol
gene_coordinates: gene coordinates start: bin start end: bin end length: bin length From length to the end, bin counts for all samples

countsj junction: annotated junction matching the current junction. gene: gene matching the current junction. strand: gene strand for the current junction in case a gene matches with the junction. multipleHit: semicolon separated list of junctions matching the current junction. symbol: gene symbol. gene_coordinates: gene coordinates. bin_spanned: semicolon separated list of all the bins spaned by this junction. j_within_bin: other junctions in the bins. From j_within_bin to the end, junction counts for all samples.

countseli event: type of event asigned by ASpli when bining. locus: gene locus locus_overlap: genes overlapping the same locus symbol: gene symbol gene_coordinates: gene coordinates start: bin start end: bin end length: bin length From length to the end, bin counts for all samples

countsie2 event: type of event asigned by ASpli when bining. locus: gene locus locus_overlap: genes overlapping the same locus symbol: gene symbol gene_coordinates: gene coordinates start: bin start end: bin end length: bin length From length to the end, bin counts for all samples

rds symbol: gene symbol locus_overlap: other genes overlapping this locus gene_coordinates: gene coordinates start: gene start end: gene end length: gene length effective_length: gene effective length From effective_length to the end, gene counts/effective_length for all samples

countsb feature: bin type event: type of event asigned by ASpli when bining. locus: gene locus locus_overlap: genes overlapping the same locus symbol: gene symbol gene_coordinates: gene coordinates start: bin start end: bin end length: bin length From length to the end, bin counts/length for all samples

condition.order The order in which ASpli is reading the conditions. This is useful for contrast tests, in order to make sure which conditions are being contrasted.

Author(s)
Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also
Accesors: countsg, countsb, countsj, countseli, countsie2, rds, rdsb, condition.order,
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_', 1:6), 
                       bam = system.file( 'extdata', bamFileNames, package="ASpli" ),
                       factor1 = c( 'C', 'C', 'C', 'D', 'D', 'D'),
                       subject = c(0, 1, 2, 0, 1, 2))

# Read counts from bam files
gbcouuts <- gbCounts( features = features, 
                      targets = targets, 
                      minReadLength = 100, maxISize = 50000, 
                      libType="SE", 
                      strandMode=0)

# Access summary and gene and bin counts and display them
gbcouuts
countsg(gbcouuts)
countsb(gbcouuts)

# Export data
writeCounts( gbcounts, output.dir = paste0(tempdir(), "/only_counts")

---

gbDUreport

**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.

**Usage**

gbDUreport( counts,
           minGenReads = 10,
           minBinReads = 5,
           minRds = 0.05,
           contrast = NULL,
           ignoreExternal = TRUE,
           ignoreIo = TRUE,
           ignoreI = FALSE,
           filterWithContrasted = TRUE,
           verbose = TRUE,
           formula = NULL,
           coef = NULL)
Arguments

- **counts**: An object of class ASpliCounts
- **minGenReads**: Genes with at least an average of minGenReads reads for any condition are included into the differential expression 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**: Either a formula or a contrast can be tested. If contrast is used, it defines 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 an 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
- **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 TRUE, 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.
- **formula**: Either a formula or a contrast can be tested. If formula is used, complex tests can be run. formula should be a formula specifying which experimental conditions defined by targets to test. If coef is specified, then that coefficient will be tested. If not, it defaults to the last term in the formula.
- **coef**: For formula only. The coefficient to be tested. If null the test defaults to the last term in the formula

Value

An ASpliDU object with results at genes, bins level.

binsDU  feature: bin type event: type of event assigned by ASpli when binning. locus: gene locus locus_overlap: genes overlapping the same locus symbol: gene symbol gene_coordinates: gene coordinates start: bin start end: bin end length: bin length logFC: bin log2 fold change between conditions pvalue: p-value bin.fdr: fdr corrected p-value for multiple testing

Author(s)
Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also
dger, jDUreport 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' ),
  subject = c(0, 1, 2, 0, 1, 2) )

# Read counts from bam files
gbcounts <- gbCounts( features = features,
  targets = targets,
  minReadLength = 100,
  maxISize = 50000,
  libType="SE",
  strandMode=0)

# Test for factor1
# Test for factor1 controlling for paired subject
gbPaired <- gbDUreport(gbcounts, formula = formula(~subject+factor1))
# Show all genes and bins ordered by FDR
genesDE(gbPaired)
binsDU(gbPaired)

# Test for factor1 without controlling for paired subject.
# Must change conditions inside gbcounts object to accommodate the contrast.
gbcards@targets$condition <- targets$factor1
gbcards@condition.order <- c("C", "D")
gbContrast <- gbDUreport(gbcards, contrast = c(1, -1))

# Show all genes and bins ordered by FDR
genesDE(gbContrast)
binsDU(gbContrast)

# Export results
writeDU( du = gbPaired, output.dir = paste0(tempdir(), "/gbPaired") )
writeDU( du = gbContrast, output.dir = paste0(tempdir(), "/gbContrast") )

---

getConditions

Retrieves condition names from a targets data frame.

**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.norm 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, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

**See Also**

DUreport.norm, DUreport.offset
integratedSignals accessors

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",
#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)

integratedSignals accessors

Accessors for ASpliIntegratedSignals object

Description

Accessors for ASpliIntegratedSignals object

Usage

signals( x )
filters( x )

Arguments

x An ASpliIntegratedSignals object

Value

Returns dataframes
**Author(s)**

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

---

**integrateSignals**

*Integrate signals*

---

**Description**

Integrates differential usage signals from different sources using overlapping regions. See vignette for more details.

**Usage**

```r
integrateSignals(sr = NULL, asd = NULL, bin.FC = 3, bin.fdr = 0.05,
nonunif = 1, usenonunif = FALSE, bin.inclusion = 0.2,
bjs.inclusion = 10.3, bjs.fdr = 0.01, a.inclusion =
0.3, a.fdr = 0.01, l.inclusion = 0.3, l.fdr = 0.01,
otherSources = NULL, overlapType = "any")
```

**Arguments**

- **sr**: An object of class ASpliSplicingReport
- **asd**: An object of class ASpliDU
- **bin.FC**: Filter bin signals by fold change. Actually, log2 fold change is returned, so default would return only bin signals with bin.fc > log2(3).
- **bin.fdr**: Filter bin signals by fdr.
- **nonunif**: Filter intronic bins with non-uniform support (nonunif < 1 is uniform)
- **usenonunif**: Use non-uniformity as filter.
- **bin.inclusion**: Filter bin signals by junction support with dPIR or dPSI accordingly.
- **bjs.inclusion**: Filter annotated junction signals by junction inclusion with dPIR or dPSI accordingly.
- **bjs.fdr**: Filter annotated junction signals by fdr.
- **a.inclusion**: Filter anchor junction signals by junction inclusion with dPIR.
- **a.fdr**: Filter anchor junction signals by fdr.
- **l.inclusion**: Filter locale junction signals by junction inclusion with dPSI.
- **l.fdr**: Filter locale junction signals by fdr.
- **otherSources**: If user wants to compare ASpli results with results from other methods, otherSources must be a GenomicRange object with all the regions found with the other methods. It will be integrated with a new column next to signals information.
- **overlapType**: Type of regions overlap matching between the different signals. Defaults to "any" and can be any of the following: "any", "start", "end", "within", "equal".
**integrateSignals**

**Value**

It returns a `ASpliIntegratedSignals` with all overlapping signals present in the region filtered by different parameters.

**Author(s)**

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

**See Also**


**Examples**

```r
# 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' ),
  subject = c(0, 1, 2, 0, 1, 2))

# Read counts from bam files
gbcounts <- gbCounts( features = features,
                        targets = targets,
                        minReadLength = 100, maxISize = 50000,
                        libType="SE",
                        strandMode=0)
jcounts <- jCounts(counts = gbcounts,
                   features = features,
                   minReadLength = 100,
                   libType="SE",
                   strandMode=0)

# Test for factor1 controlling for paired subject
gbPaired <- gbDUreport(gbcounts, formula = formula(~subject+factor1))
jPaired <- jDUreport(jcounts, formula = formula(~subject+factor1))

# Generate a splicing report merging bins and junctions DU
```
sr <- splicingReport(gbPaired, jPaired, gbcounts)
is <- integrateSignals(sr, jcounts)

# Show integrate signals results and filters used
signals(is)
filters(is)

jCounts

Report PSI, PJU and PIR using experimental junctions

Description

Summarize read overlaps against junctions. Report PSI, PJU, PIR and counts for experimental junctions. PSI or PIR metrics are 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

jCounts( counts, features, minReadLength,
threshold = 5, minAnchor = 10, libType="SE",
strandMode=0, alignFastq = FALSE, dropBAM = FALSE )

Arguments

counts An object of class ASpliCounts.
features An object of class ASpliFeatures.
minReadLength Minimum read length of sequenced library. It is used for computing E1I and IE2 read summarization. Make sure this number is smaller than the maximum read length in every bam file, otherwise no junctions will be found.
threshold Minimum number of reads supporting junctions. Default=5
minAnchor An intronic junction must overlap completely and at least an minAnchor% into the exon region and the intron region. The regions can be exon1-intron or intron-exon2.
libType Defines how reads will be treated according their sequencing library type (paired (PE, default) or single end (SE))
strandMode controls the behavior of the strand getter. It indicates how the strand of a pair should be inferred from the strand of the first and last alignments in the pair. 0: strand of the pair is always *. 1: strand of the pair is strand of its first alignment. This mode should be used when the paired-end data was generated using one of the following stranded protocols: Directional Illumina (Ligation), Standard SOLiD. 2: strand of the pair is strand of its last alignment. This mode should be used when the paired-end data was generated using one of the following stranded protocols: dUTP, NSR, NNSR, Illumina stranded TruSeq PE protocol. For more information see ?strandMode
**alignFastq**

Experimental (that means it’s highly recommended to leave the default, FALSE): executes an alignment step previous to Bam summarization. Useful if not enough space on local disks for beans so fasts can be aligned on the fly, even from a remote machine, and then the BAMs can be deleted after each summarization. If set to TRUE, targets data frame must have a column named alignerCall with complete call to aligner for each sample. ie: STAR –runMode alignReads –outSAMtype BAM SortedByCoordinate –readFilesCommand zcat –genomeDir /path/to/STAR/genome/folder -runThreadN 4 -outFileNamePrefix sample name –readFilesIn /path/to/R1 /path/to/R2. Output must match bam files provided in targets.

**dropBAM**

Experimental (that means it’s highly recommended to leave the default, FALSE): If alignFastq is TRUE, deletes BAMs after summarization. Used in conjunction with alignFastq to delete BAMs when there’s not enough free space on disk. Use with caution as it will delete all files in "bam" column in targets dataframe.

**Value**

An object of class ASpliAS. Accessors: irPIR, esPSI, altPSI, junctionsPIR, junctionsPJU

**irPIR**

event: Type of event assigned by ASpli when binning. J1: Semicolon separated list of all the junctions with an end matching the start of the intron. J2: Semicolon separated list of all the junctions with an end matching the end of the intron. J3: Semicolon separated list of all the junctions overlapping the intron. All the columns from J1 to J2 represent the J1 counts in the different samples for each bin. The counts are the sum of all the J1 junctions. All the columns from J2 to J3 represent the J2 counts in the different samples for each bin. The counts are the sum of all the J2 junctions. All the columns from J3 to the first condition represent the J3 counts in the different samples for each bin. The counts are the sum of all the J3 junctions. The last columns are the PIR metrics calculated for each condition. The PIR metric is calculated as:

\[
P_{IR} = \frac{J_1 + J_2}{J_1 + J_2 + 2 \times J_3}
\]

Where the junctions are the sum by condition.

**altPSI**

event: Type of event assigned by ASpli when binning. J1(J2): Semicolon separated list of all the junctions with an end matching the end of alt5’SS(alt3’SS). J3: Semicolon separated list of all the junctions with an end matching the start of alt5’SS or the start of alt3’SS. All the columns from J1 to J2 represent the J1 counts in the different samples for each bin. The counts are the sum of all the J1 junctions. All the columns from J2 to J3 represent the J2 counts in the different samples for each bin. The counts are the sum of all the J2 junctions. All the columns from J3 to the first condition represent the J3 counts in the different samples for each bin. The counts are the sum of all the J3 junctions. The last columns are the PSI metrics calculated for each condition. The PSI metric is calculated as:

\[
PSI = \frac{J_{12}}{J_{12} + J_3}
\]

Where J12 is J1 if it’s an alt 5’ event or J2 if it’s an alt 3’ event and the junctions are the sum by condition.
event: Type of event assigned by ASpli when binning J1: Semicolon separated list of all the junctions with an end on the alternative exon. J2: Semicolon separated list of all the junctions with an end on the alternative exon. J3: Semicolon separated list of all the junctions overlapping the alternative exon. All the columns from J1 to J2 represent the J1 counts in the different samples for each bin. The counts are the sum of all the J1 junctions. All the columns from J2 to J3 represent the J2 counts in the different samples for each bin. The counts are the sum of all the J2 junctions. All the columns from J3 to the first condition represent the J3 counts in the different samples for each bin. The counts are the sum of all the J3 junctions. The PSI metric is calculated as:

\[ PSI = \frac{J_1 + J_2}{J_1 + J_2 + 2 \times J_3} \]

Where the junctions are the sum by condition.

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. hitIntron: If the junction matches a bin, the bin is shown here. hitIntronEvent: If the junction matches a bin, the type of event assigned by ASpli to this bin. All the columns from hitIntronEvent up to the first repetition of the samples names in the columns, represent the J1 counts in the different samples for each region. From there to the next time the names of the columns repeat themselves, the J2 counts and from there to the first condition, the J3 counts. The last columns are the PIR metrics calculated for each condition. The PIR metric is calculated as:

\[ PIR = \frac{J_1 + J_2}{J_1 + J_2 + 2 \times J_3} \]

Where the junctions are the sum by condition.

Given a junction, it is possible to analyze if it shares start, end or both with another junction. If so, it is because there is alternative splicing. Junction: name of the junction. gene: gene it belongs to. strand: gene strand. multipleHit: if other gene overlaps the gene the junction belongs to. symbol: gene symbol. gene_coordinates: gene coordinates. bin_spanned: semicolon separated list of all the bins spaned by this junction. j_within_bin: other junctions in the bins. StartHit: all the junctions sharing the start with this junction and $SPJU_{J1} = J3/(J1 + J3)$ for each condition. EndHit: all the junctions sharing the end with this junction and $SPJU_{J2} = J3/(J2 + J3)$ for each condition. All the columns between j_within_bin and StartHit are the counts for J3 in the different samples for each region. From there to EndHit, the J1 counts and $SPJU_{J1} = J3/(J1 + J3)$ for each condition. Then after EndHit, the J2 counts and $SPJU_{J2} = J3/(J2 + J3)$. Rownames are J3 range. StartHit is J1 range and EndHit is J2 range.

Author(s)

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

See Also

Accessors: irPIR, altPSI, esPSI, junctionsPIR, junctionsPJU

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_',1:6),
  bam = system.file('extdata', bamFileNames, package="ASpli" ),
  factor1 = c('C','C','C','D','D','D'),
  subject = c(0,1,2,0,1,2))

# Read counts from bam files
gbcounts <- gbCounts( features = features, 
  targets = targets, 
  minReadLength = 100, maxISize = 50000, 
  libType="SE", 
  strandMode=0)

jcounts <- jCounts(counts = gbcounts, 
  features = features, 
  minReadLength = 100, 
  libType="SE", 
  strandMode=0)

# Access summary and gene and bin counts and display them
gbcounts
countsg(gbcounts)
countsb(gbcounts)

# Access summary and junction counts and display them
jcounts
irPIR(jcounts)
esPSI(jcounts)
altPSI(jcounts)
junctionsPIR(jcounts)
junctionsPJU(jcounts)
```r
# Export data
writeAS(as = jcounts, output.dir = paste0(tempdir(), "/only_as" ))
```

## JDU accessors

### Accessors for ASpliJDU object

#### Description

Accessors for ASpliJDU object

#### Usage

- `anchorc(x)`
- `anchorj(x)`
- `localec(x)`
- `localej(x)`
- `jir(x)`
- `jes(x)`
- `jalt(x)`

#### Arguments

- `x` An ASpliJDU object

#### Value

Returns dataframes

#### Author(s)

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

---

### Description

This function estimates the differential usage of junctions combining different types of evidence

Differential junction usage is estimated using a combination of evidences
Usage

```
jdUreport(asd,
  minAvgCounts = 5,
  contrast = NULL,
  filterWithContrasted = TRUE,
  runUniformityTest = FALSE,
  mergedBams = NULL,
  maxPValForUniformityCheck = 0.2,
  strongFilter = TRUE,
  maxConditionsForDispersionEstimate = 24,
  formula = NULL,
  coef = NULL,
  maxFDRForParticipation = 0.05,
  useSubset = FALSE)
```

Arguments

- **asd**: An object of class `ASpliAS` with results of PSI and PIR using experimental junctions

- **minAvgCounts**: Minimum average counts for filtering

- **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. If NULL must provide a formula.

- **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 TRUE, it is strongly recommended to do not change this value.

- **runUniformityTest**: Run uniformity test on Intron Retention. Sometimes Mutually Exclusive Exons (MEX) events can be confused with Intron Retention events. This test compares the standard deviation of the inner intron region (11 bases from both ends) to the mean of both intron ends. Numbers closer to 0 mean the event is more probably an Intron Retention event than an MEX event. The test takes some time to run so it defaults to FALSE.

- **mergedBams**: Path to merged bams for each testing condition. If no merged bams exist (for example, paired samples without replicates), use the same bams as targets.

- **maxPValForUniformityCheck**: To speed up uniformity test only check junctions with pval < maxPValForUniformityCheck
strongFilter  If strongFilter is TRUE, then we remove all events with at least one junction that doesn't pass the filter.

maxConditionsForDispersionEstimate  In order to reduce resource usage, estimate dispersion for statistics tests with a reduced number of conditions.

formula  Either a formula or a contrast can be tested. If formula is used, complex tests can be run. formula should be a formula specifying which experimental conditions defined by targets to test. If coef is specified, then that coefficient will be tested. If not, it defaults to the last term in the formula.

toef  For formula only. The coefficient to be tested. If null the test defaults to the last term in the formula

maxFDRForParticipation  In order to calculate junctionPSI participation, only use significant junctions (ie junctions with FDR < maxFDRForParticipation).

useSubset  Experimental. It is strongly recommended to leave the default, FALSE.

Details

Estimation is made at junction level using diffSpliceDGE function from edgeR package. Junctions belonging to the same AS event comprises the event "set". Each junction is tested against this "set" in a similar fashion that bins are tested against their gene in diffSpliceDGE. Localec are clusters made of junctions that share an end with at least another junction in the cluster.

Value

An ASpliJDU object with results of differential usage at junctions level.

localec  size: number of junctions belonging to the cluster. cluster.LR: likelihood ratio of cluster differential usage. pvalue: pvalue of cluster differential usage. FDR: fdr of cluster differential usage. range: cluster location. participation: participation of the significant junction (FDR < maxFDRForParticipation) presenting maximal participation value inside the cluster dParticipation: delta participation of the significant junction (FDR < maxFDRForParticipation) presenting maximal participation value inside the cluster

localej  cluster: name of the cluster the junction belongs to log.mean: log of mean counts across all conditions for this junction logFC: log fold change of junction across conditions pvalue: pvalue of junction FDR: FDR of junction annotated: is junction annotated or new participation: the maximal participation value observed across contrasted conditions dParticipation: delta participation of the maximal participation value observed across contrasted conditions From dParticipation to the end, junction counts for all samples


anchorj  log.mean: log of mean counts across all conditions for this junction logFC: log fold change of junction across conditions LR: likelihood ratio of junction differential usage. pvalue: pvalue of junction FDR: FDR of junction J1.pvalue: pvalue of J1 junction J2.pvalue: pvalue of J2 junction NonUniformity: if non
uniformity test was performed, numbers closer to zero mean uniformity and
closer to one mean non uniformity dPIR: junction delta PIR annotated: is junc-
tion annotated or new From annotated to the end, junction counts for all samples

jir
J3: J3 junction/s logFC: log fold change of junction accross conditions log.mean:
log of mean counts accross all conditions for this junction pvalue: pvalue of
junction FDR: FDR of junction LR: likelihood ratio of junction differential usage. NonUniformity: if non uniformity test was performed, numbers closer to
zero mean uniformity and closer to one mean non uniformity dPIR: junction
delta PIR multiplicity: do multiple junctions cross the region From multiplicity
to the end, junction counts for all samples

jes
event: type of event J3: J3 junction/s logFC: log fold change of junction accross
conditions log.mean: log of mean counts accross all conditions for this junc-
tion pvalue: pvalue of junction FDR: FDR of junction LR: likelihood ratio of
junction differential usage. dPSI: junction delta PSI multiplicity: do multiple
junctions cross the region From multiplicity to the end, junction counts for all
samples

jalt
event: type of event J3: J3 junction/s logFC: log fold change of junction accross
conditions log.mean: log of mean counts accross all conditions for this junc-
tion pvalue: pvalue of junction FDR: FDR of junction LR: likelihood ratio of
junction differential usage. dPSI: junction delta PSI multiplicity: do multiple
junctions cross the region From multiplicity to the end, junction counts for all
samples

contrast
Conditions contrasted by ASpli

Author(s)
Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also
Accessors: localec, localej, anchorc, anchorj, jir, jes, jalt, junctionsDU, Export: writeJDU,
writeDU, edgeR, 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" )
targets <- data.frame(
  row.names = paste0('Sample_ ', 1:6),
  bam = system.file('extdata', bamFileNames, package="ASpli"),
  factor1 = c('C', 'C', 'C', 'D', 'D', 'D'),
  subject = c(0, 1, 2, 0, 1, 2))

# Read counts from bam files
gbcoungs <- gbCounts(features = features,
  targets = targets,
  minReadLength = 100, maxISize = 50000,
  libType="SE",
  strandMode=0)
jcounts <- jCounts(counts = gbcounts,
  features = features,
  minReadLength = 100,
  libType="SE",
  strandMode=0)

# Test for factor1 controlling for paired subject
jPaired <- jDUreport(jcounts, formula = formula(~subject+factor1))

# Show junctions information
jPaired
localej(jPaired)
localec(jPaired)
anchorj(jPaired)
anchorc(jPaired)
jir(jPaired)
jes(jPaired)
jalt(jPaired)

# Export results
writeJDU(jPaired, output.dir = paste0(tempdir(), "/jPaired") )

---

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)

Arguments

counts An object of class ASpliCounts

targets A dataframe containing sample, bam and experimental factor columns.

appendTo An object of class ASpliDU to which append the results of junction differential usage. If appendTo is NULL a new ASpliDU is created

minGenReads Junctions within genes with at least an average of minGenReads reads for all conditions are included into the differential junction usage test. Default value is 10 reads.

minRds Junctions within genes with at least an average of minRds read density for all conditions are included into the differential bin usage test. Junctions with at least an average of minRds read density for any condition are included into the differential junction usage test. Default value is 0.05.

threshold Junction with at least threshold counts are included into the differential usage test.

offset Corrects junction counts using an offset matrix derived from gene expression data. Default = FALSE

offsetUseFitGeneX Fit a GLM using gene counts to build the offset matrix. This argument is used only when 'offset' argument is set to TRUE. The default value is 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 an 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

Value

An ASpliDU object with results of differential usage of junctions

Author(s)

Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz
See Also

edgeR, DUreport

Accessors: junctionsDU

Export: writeDU

Examples

# This function has been deprecated. Please see vignette for new pipeline.

---

loadBAM

*Load BAM files*

**Description**

Load BAM files into R session using a targets specification.

**Usage**

```r
loadBAM(targets, cores, libType, strandMode)
```

**Arguments**

- `targets`: A data frame containing sample, bam and experimental factors columns
- `cores`: Number of processors to use
- `libType`: Options are: "SE" or "PE"
- `strandMode`: Options are: 0,1,2. See ?strandMode for more information

**Value**

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

**Author(s)**

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

**Examples**

```r
# 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. 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 an 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 NULL.

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, Andres Rabinovich, 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', 'B', 'B', 'B', 'B', 'B', 'B', 'B'),
# factor2 = c( 'C', 'C', 'C', 'D', 'D', 'D', 'C', 'C', 'D', '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.norm( 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) )

---

**plotBins**  
*Draw plots of gene counts, bin counts, PSI/PIR value, inclusion and exclusion junctions for selected bins.*

**Description**

Creates a plot with gene counts, bin counts, PSI/PIR value, inclusion and exclusion junctions for selected bins and conditions.
plotBins

Usage

plotBins( counts, as, bin, factorsAndValues, targets, main = NULL, colors = c( '#2F7955', '#79552F', '#465579', '#A04935', '#752020', '#A07C35' ), panelTitleColors = '#000000', panelTitleCex = 1, innerMargins = c( 2.1, 3.1, 1.1, 1.1 ), outerMargins = c( 0, 0, 2.4, 0 ), useBarplots = NULL, barWidth = 0.9, barSpacer = 0.4, las.x = 2, useHCColors = FALSE, legendAtSide = TRUE, outfolder = NULL, outfileType = c( 'png', 'bmp', 'jpeg', 'tiff', 'pdf' )[1], deviceOpt = NULL )

Arguments

counts An object of class ASplitCounts
as An object of class ASplitAS
bin A character vector with the names of the bins to be plotted.
factorsAndValues A list containing the factor and the values for each factor to be plotted. The order of the factors will modify how the conditions are grouped in the plot. factorsAndValues must be a named list, where the name of each element is a factor and the list element itself is a character vector of the values of this factor in the order to be plotted. See examples for more details.
targets A data frame containing sample, bam files and experimental factor columns
main Main title of the plot. If NULL the bin name is used as title.
colors A vector of character colors for lines and bar plots.
panelTitleColors A vector of character colors for the titles of each plot panel.
panelTitleCex Character size expansion for panel titles.
innerMargins A numerical vector of the form c(bottom, left, top, right) which gives the size of each plot panel margins. Defaults to c( 2.1, 3.1, 1.1, 1.1 )
outerMargins A numerical vector of the form c(bottom, left, top, right) which gives the size of margins. Defaults to c( 0, 0, 2.4, 0 )
useBarplots  A logical value that indicates the type of plot to be used. If TRUE bar plots are used, if FALSE lines are used. If NULL the type is bar plot if there just two conditions and lines if there are more than two conditions.

barWidth  The width of the bars in bar plots. barWidth must be in (0,1] range. Default value is 0.9.

barSpacer  Fraction of barwidth used as spacer between bar plot groups. Default value is 0.4.

las.x  Text orientation of x-axis labels.

useHCColors  A logical value. If TRUE panelTitleColors are not used, instead panel title are automatically chosen to have high contrast against colors.

legendAtSide  A logical value that forces panel title to be shown on the y-axis, instead of over the plot.

outfolder  Path to output folder to write plot images. Is NULL, plot are rendered on the default device.

outFileType  File format of the output files used if outfolder is not NULL. Accepted values are ‘png’, ‘jpeg’, ‘tiff’, ‘pdf’. Each value selects the graphic device of the same name. The name of the image file is the name of bin with the corresponding extension given by the chosen type.

deviceOpt  A list of named options to be passed to the graphic device selected in outFileType.

Value

Returns a png for each selected bin.

Author(s)

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

See Also

plotGenomicRegions.

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

# Plot bin data. Factor2 is the main factor for graphic representation in
# this example as it is the first in factorsAndValues argument.
# This makes a bar plot comparing four conditions, grouped by factor1.
#plotBins( counts, as, 'GENE03:E002',
# factorsAndValues = list( #
# factor2 = c('C','D'),
# factor1 = c('A','B') ),
# las.x = 1,
# legendAtSide = TRUE,
# useHCColors = TRUE,
# targets = targets,
# barWidth = 0.95,
# innerMargins = c( 2.1, 4.1, 1.1, 1.1 ) )

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

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

# This makes a line plot for six conditions, grouped by factor1.
#plotBins( counts, as, 'GENE03:E002',
# factorsAndValues = list( #
# factor1 = c('A','B','C','D','E','F' ) ),
# las.x = 1,
# legendAtSide = FALSE,
# targets = targets,
plotGenomicRegions

Create genomic regions coverage plots

Description

Graphic representation of coverage and junctions is useful to complement the results of differential usage of bins and junction and differential expression analysis. Function plotGenomicRegions allow to create plots for multiple conditions for bins and genes. Each individual plot can only correspond to a single gene or bin, but can contain many panels for different experimental conditions.

Usage

plotGenomicRegions( features, x, genomeTxDb, targets, xIsBin = TRUE, layout = 'auto', colors = 'auto', plotTitles = 'auto', sashimi = FALSE, zoomOnBins = FALSE, deviceOpt = NULL, highLightBin = TRUE, outfolder = NULL, outfileType = 'png', mainFontSize = 24, annotationHeight = 0.2, annotationCol = 'black', annotationFill = 'gray', annotationColTitle = 'black', preMergedBAMs = NULL, useTransparency = FALSE, tempFolder = 'tmp', avoidReMergeBams = FALSE, verbose = TRUE )

Arguments

features      An ASpliFeatures object, generated with binGenoms function.
x            A character vector with the names of bins or genes to plot. To plot into a window is recommended that the length of x be one.
genomeTxDb        A TxDb object with the annotation of reference genome.
targets          A data frame containing sample, bam files and experimental factor columns
xIsBin          A logical value that indicates if values in x corresponds to gene names or bin names.
layout            A character with value 'auto' or a character matrix with condition names arranged with the desired layout of the panels in plots. The dimensions of layout matrix, colors matrix and plotTitles matrices must be the same. Matrix can have NA values, however, the height of the panels corresponding to that column are modified to occupy the total height. The default value is 'auto'.
colors         A character containing value 'auto' or containing colors strings, or a character matrix with color names arranged with the layout specified in layout argument. The dimensions of layout matrix, colors matrix and plotTitles matrices must be the same. The default value is 'auto'.
plotTitles        A character containing value 'auto', or a character matrix with titles for each panel arranged with the layout specified in layout argument. The dimensions of layout matrix, colors matrix and plotTitles matrices must be the same. The default value is 'auto'.

# innerMargins = c( 2.1, 4.1, 1.1, 1.1 )

sashimi  A logical value that specifies that a sashimi plot for junctions must be included into each panel. The default value is FALSE.

zoomOnBins  A FALSE logical value or a double value between 0 and 1. If value is FALSE then the genomic range to be plotted correspond to the complete gene, otherwise the genomic range is that the size of the bin corresponds to a fraction equals to zoomOnBins value of the total and is centered in the bin. Is used only when xIsBin is TRUE. The default value is FALSE.

deviceOpt  A named list of arguments to be passed to the graphic device used to plot. This allow to further customization of the plot. The default value is an empty list.

highlightBin  A logical value that indicates if the bin should be highlighted. The default value is TRUE.

outfolder  NULL or a character vector representing a folder path that will be used to save the plot images. If the folder doesn’t exists it is created. If NULL, the plot is made in a window. The default value is NULL. For each bin, a single image file is generated. The name of the file is the name of the bin, added with a ‘.gr.’ string, and the file extension at the end. If the name of the bin contains invalid character for a file name, those will be replaced by an underscore character.

outFileType  A character value the specifies the file format of the plot to be created. Is used only when outfolder is not NULL. Accepted values are ‘png’, ‘jpeg’, ‘bmp’, ‘tiff’, ‘pdf’. Each value is the graphic device used to create the image. The default value is ‘png’.

mainFontSize  A numeric value specifying the size of the main title. The default value is 24.

annotationHeight  A double value specifying the proportion of the total height used to represent the gene model. The default value is 0.2.

annotationCol  A character value that specifies the color of the borders of bars in gene model representation. The default value is ‘black’.

annotationFill  A character value that specifies the color of the filling of bars in gene model representation. The default value is ‘gray’.

annotationColTitle  A character value that specifies the color of text in gene model representation. The default value is ‘black’.

preMergedBAMs  A one column data frame that associates a condition, specified in the row name, with a character value representing the path of bam file with the reads for that condition. This bam file is typically generated by merging the bam files of all replicates for that condition. The default value is NULL, this specifies that not merged bam files are used, instead on-the-fly read extraction and merging is done from the bam files specified in the targets argument.

useTransparency  A logical value that specifies if transparency will be used to generate the plots, this leads to better looking plot, however not all graphic devices support transparency. The default value is FALSE.

tempFolder  A character value specifying the path to store intermediate files produced while extracting and merging reads from bam files. It is only used when preMerged-BAMs arguments is NULL. The files created are not automatically removed.
after plotting because can be reused to create a new plot of the same genes or bins with different graphic options. The default value is 'tmp', that means a new 'tmp' folder in the current working folder.

**avoidReMergeBams**
A logical value specifying that extraction and merging of bam files will be avoided. This is only meaningful when the extraction and merging of the same set of genes and bins was done in the previous execution of `plotGenomicRegions` function. The default value is FALSE.

**verbose**
A logical value specifying that detailed information about the execution will be informed to the user. The default value is TRUE.

**Value**

Returns a png for each selected bin

**Author(s)**

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

**See Also**

* Devices, pdf, png, bmp, jpeg, tiff

**Examples**

```r
# 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'),
# stringsAsFactors = FALSE )

# Plot a single bin to a window
#plotGenomicRegions( #
# features,
# 'GENE01:E002',
# genomeTxDb,
# targets,
# sashimi = FALSE,
# colors = '#AA4444',
```

---

`plotGenomicRegions` function in the package ASpli is a tool for visualizing genomics data. It allows for the creation of plots from genomic regions defined by a transcript database and BAM files. The function provides options to avoid merging bam files and to control verbose output.

The `avoidReMergeBams` parameter is crucial when you want to ensure that the extraction and merging of the same set of genes and bins are not repeated unnecessarily. This can be particularly useful in long-running processes where merging can take a significant amount of time.

The `verbose` parameter is equally important as it controls the level of detailed information that the function provides during its execution. Setting it to TRUE ensures that you receive detailed feedback about each step of the process, which can be invaluable for debugging and monitoring.

The `Value` section clarifies that the function returns a PNG file for each selected bin, which is a common output format for visual data representations.

The `Author(s)` section lists the contributors to this function, which can be informative for users looking to acknowledge contributions or to reach out to authors for more information.

The `See Also` section provides links to other packages or functions that may be useful in conjunction with `plotGenomicRegions`, such as `Devices` for different output formats.

The `Examples` section includes a code snippet that demonstrates how to use `plotGenomicRegions` to plot a single bin. This example would typically be part of the documentation to show users how to apply the function in practice.
Read density of gene and bins

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.
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, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz
splicingReport

Description
This function integrates bin and junction usage in a comprehensive report.

Usage

splicingReport(bdu, jdu, counts)

Arguments

bdu An object of class ASpliDU
jdu An object of class ASpliJDU
counts An object of class ASpliCounts

Value

Author(s)
Andres Rabinovich, Estefania Mancini, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also

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'),
subject = c(0, 1, 2, 0, 1, 2))

# Read counts from bam files
gbcounts <- gbCounts( features = features,
    targets = targets,
    minReadLength = 100, maxISize = 50000,
    libType="SE",
    strandMode=0)
jcounts <- jCounts(counts = gbcounts,
    features = features,
    minReadLength = 100,
    libType="SE",
    strandMode=0)

# Test for factor1 controlling for paired subject
gbPaired <- gbDUreport(gbcounts, formula = formula(~subject+factor1))
jPaired <- jDUreport(jcounts, formula = formula(~subject+factor1))

# Generate a splicing report merging bins and junctions DU
sr <- splicingReport(gbPaired, jPaired, gbcounts)

# Access splicing report elements
sr
localebased(sr)
anchorbased(sr)
binbased(sr)

splicingReport accessors

Accessors for ASpliSplicingReport object

Description
Accessors for ASpliSplicingReport object

Usage

binbased( x )
localebased( x )
anchorbased( x )

Arguments

x An ASpliSplicingReport object

Value

Returns dataframes
Subset ASpli objects

Author(s)
Estefania Mancini, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

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
x An ASpliCount or ASpliAS object for subset function, or list of GAlignments for subsetBams function.
targets A dataframe containing sample, bam and experimental factor columns.
select 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.
removeRedundantExpFactors 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.
...
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, Andres Rabinovich, 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 )
#
# 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")
writeJDU(jdu, output.dir="jdu")
writeSplicingReport(sr, output.dir="sr")
writeAll(counts, du, as, output.dir="output")

Arguments

counts     An ASpliCounts object
as         An ASpliAS object
du         An ASpliDU object
jdu        An ASpliJDU object
sr         An ASpliSplicingReport 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, Andres Rabinovich, Javier Iserte, Marcelo Yanovsky, Ariel Chernomoretz

See Also

jCounts, binGenome, DUreport.norm, DUreport.offset

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

See Also

jCounts, binGenome, DUreport.norm, DUreport.offset
Index

* RNA-seq alternative splicing analysis using bin coverage and junctions
  ASpli-package, 3
  altPSI, 7, 45
  altPSI (AS accessors), 4
  altPSI, ASpliAS-method (ASpliAS-class), 6
  altPSI<-(AS accessors), 4
  altPSI<-, ASpliAS, data.frame-method (ASpliAS-class), 6
  anchorbased, 63
  anchorbased (splicingReport accessors), 64
  anchorbased, ASpliSplicingReport-method (ASpliSplicingReport-class), 11
  anchorbased<-(splicingReport accessors), 64
  anchorbased<-, ASpliSplicingReport, data.frame-method (ASpliSplicingReport-class), 11
  anchorc, 49
  anchorc (JDU accessors), 46
  anchorc, ASpliJDU-method (ASpliJDU-class), 10
  anchorc<-(JDU accessors), 46
  anchorc<-, ASpliJDU, data.frame-method (ASpliJDU-class), 10
  anchorj, 49
  anchorj (JDU accessors), 46
  anchorj, ASpliJDU-method (ASpliJDU-class), 10
  anchorj<-(JDU accessors), 46
  anchorj<-, ASpliJDU, data.frame-method (ASpliJDU-class), 10
  AS accessors, 4
  AsDiscover, 7
  AsDiscover (jCounts), 42
  AsDiscover, ASpliCounts-method (ASpliCounts-class), 8
  ASpli (ASpli-package), 3
  ASpli-deprecated, 6
  ASpliAS, 49, 53
  ASpliAS (ASpliAS-class), 6
  ASpliAS-class, 6
  aspliASexample (Example data), 24
  aspliBamsExample (Example data), 24
  ASpliCounts, 7
  ASpliDU, 53
  ASpliDU (ASpliDU-class), 9
  ASpliDU-class, 9
  aspliDUexample1 (Example data), 24
  aspliDUexample2 (Example data), 24
  aspliExampleBamList (Example data), 24
  aspliExampleGTF (Example data), 24
  ASpliFeatures-class, 9
  aspliFeaturesExample (Example data), 24
  ASpliIntegratedSignals, 26, 41
  ASpliIntegratedSignals (ASpliIntegratedSignals-class), 10
  ASpliIntegratedSignals-class, 10
  ASpliJDU (ASpliJDU-class), 10
  ASpliJDU-class, 10
  aspliJunctionDUexample (Example data), 24
  ASpliSplicingReport, 26, 27, 32, 41, 63
  ASpliSplicingReport (ASpliSplicingReport-class), 11
  ASpliSplicingReport-class, 11
  aspliTargetsExample (Example data), 24
  binbased, 63
  binbased (splicingReport accessors), 64
  binbased, ASpliSplicingReport-method (ASpliSplicingReport-class), 11
  binbased<-(splicingReport accessors), 64
INDEX

binbased<-, ASpliSplicingReport, data.frame-method
  (ASpliSplicingReport-class), 11
binGenome, 11, 67
binGenome, TxDb-method
  (binGenome-methods), 13
binGenome-methods, 13
binsDU, 17, 19, 21, 23, 37
binsDU, ASpliDU-method (ASpliDU-class), 9
binsDU<-, (DU accessors), 14
binsDU<-, ASpliDU-method
  (ASpliDU-class), 9
bump, 60
countsb (Counts accesors), 13
countsb (Counts accesors), 13
countsb, ASpliCounts-method
  (ASpliCounts-class), 8
countsb<-, (Counts accesors), 13
countsb<-, ASpliCounts, data.frame-method
  (ASpliCounts-class), 8
countsie1, 13
countsie1<-, (Counts accesors), 13
countsie1<-, ASpliCounts, data.frame-method
  (ASpliCounts-class), 8
counts interleaved 1, 13
counts interleaved 1, ASpliCounts-method
  (ASpliCounts-class), 8
counts interleaved 1<-, (Counts accesors), 13
counts interleaved 1<-, ASpliCounts, data.frame-method
  (ASpliCounts-class), 8
countsj, 34
countsj<-, (Counts accesors), 13
countsj<-, ASpliCounts, data.frame-method
  (ASpliCounts-class), 8
counts interleaved 2, 13
counts interleaved 2, ASpliCounts-method
  (ASpliCounts-class), 8
counts interleaved 2<-, (Counts accesors), 13
counts interleaved 2<-, ASpliCounts, data.frame-method
  (ASpliCounts-class), 8
devices, 60
devices, DU accessors, 14
dUreport, 15, 52
dUreport, ASpliCounts-method
  (ASpliCounts-class), 8
dUreport.norm, 17, 30, 38, 67
dUreport.norm, ASpliCounts-method
  (ASpliCounts-class), 8
dUreport.offset, 19, 30, 38, 67
dUreport.offset, ASpliCounts-method
  (ASpliCounts-class), 8
dUreportBinSplice, 21
dUreportBinSplice, ASpliCounts-method
  (ASpliCounts-class), 8
dgeR, 17, 19, 21, 23, 37, 49, 52
esPSI, 7, 45
esPSI, AS accessors, 4
esPSI, ASpliAS-method (ASpliAS-class), 6
esPSI<-, AS accessors, 4
esPSI<-, ASpliAS, data.frame-method
  (ASpliAS-class), 6
features, Examine ASpliDU objects, 23
Example data, 24
exportIntegratedSignals, 6, 25, 41
exportIntegratedSignals, ASpliIntegratedSignals-method
  (ASpliIntegratedSignals-class), 10
exportSplicingReports, 6, 27
exportSplicingReports, ASpliSplicingReport-method
  (ASpliSplicingReport-class), 11
features, 28
features interleaved, 13
features, features interleaved, 28
featuresb, ASpliFeatures-method
(ASpliFeatures-class), 9
featuresb<-(Features accessors), 28
featuresb<-, ASpliFeatures, GRanges-method
(ASpliFeatures-class), 9
featuresg, 12, 13
featuresg (Features accessors), 28
featuresg<-, ASpliFeatures-method
(ASpliFeatures-class), 9
featuresg<-, ASpliFeatures, GRanges-method
(ASpliFeatures-class), 9
featuresj, 12, 13
featuresj (Features accessors), 28
featuresj<-, ASpliFeatures-method
(ASpliFeatures-class), 9
featuresj<-, ASpliFeatures, GRanges-method
(ASpliFeatures-class), 9
filterDU, 29
filterDU, ASpliDU-method
(ASpliDU-class), 9
filters, 41
filters (integratedSignals accessors), 39
filters, ASpliIntegratedSignals-method
(ASpliIntegratedSignals-class), 10
filters<-, integratedSignals accessors), 39
filters<-, ASpliIntegratedSignals-method,
(ASpliIntegratedSignals-class), 10
filterSignals, 31
filterSignals, ASpliSplicingReport-method
(ASpliSplicingReport-class), 11

gbCounts, 6, 32
gbCounts, ASpliFeatures-method
(ASpliFeatures-class), 9
gbDUreport, 6, 26, 27, 30, 35, 41, 63
gbDUreport, ASpliCounts-method
(ASpliCounts-class), 8
genesDE, 17, 19, 21, 23, 37
genesDE (DU accessors), 14
genesDE, ASpliDU-method (ASpliDU-class), 9
genesDE<-, DU accessors), 14
genesDE<-, ASpliDU, data.frame-method
(ASpliDU-class), 9
getConditions, 38
integratedSignals accessors, 39
integrateSignals, 6, 40
integrateSignals, ASpliSplicingReport-method
(ASpliSplicingReport-class), 11
irPIR, 7, 45
irPIR (AS accessors), 4
irPIR, ASpliAS-method (ASpliAS-class), 6
irPIR<-, ASpliAS, data.frame-method
(ASpliAS-class), 6
jalt, 49
jalt (JDU accessors), 46
jalt, ASpliJDU-method (ASpliJDU-class), 10
jalt<-, JDU accessors), 46
jalt<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
jCounts, 6, 42, 67
jCounts, ASpliCounts-method
(ASpliCounts-class), 8
JDU accessors, 46
jDUREport, 6, 19, 21, 26, 27, 30, 37, 41, 46, 63
jDUREport, ASpliAS-method
(ASpliAS-class), 6
jDUREport, ASpliJDU-method
(ASpliJDU-class), 10
jes, 49
jes (JDU accessors), 46
jes, ASpliJDU-method (ASpliJDU-class), 10
jes<-, JDU accessors), 46
jes<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
jir, 49
jir (JDU accessors), 46
jir, ASpliJDU-method (ASpliJDU-class), 10
jir<-, JDU accessors), 46
jir<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
joint, 7
joint (AS accessors), 4
joint, ASpliAS-method (ASpliAS-class), 6
joint<-, ASpliAS, data.frame-method
(ASpliAS-class), 6
jpeg, 60
junctionDUreport, 17, 23, 50
junctionDUreport, ASpliCounts-method
(ASpliCounts-class), 8
junctionsDU, 49, 52
junctionsDU (DU accessors), 14
junctionsDU, ASpliDU-method
(ASpliDU-class), 9
junctionsDU<-(DU accessors), 14
junctionsDU<-, ASpliDU, data.frame-method
(ASpliDU-class), 9
junctionsPIR, 7, 45
junctionsPIR (AS accessors), 4
junctionsPIR, ASpliAS-method
(ASpliAS-class), 6
junctionsPIR<-, (AS accessors), 4
junctionsPIR<-, ASpliAS, data.frame-method
(ASpliAS-class), 6
loadBAM, 52
localebased, 63
localebased (splicingReport accessors), 64
localebased, ASpliSplicingReport-method
(ASpliSplicingReport-class), 11
localebased<-(splicingReport accessors), 64
localebased<-, ASpliSplicingReport, data.frame-method
(ASpliSplicingReport-class), 11
localec, 49
localec (JDU accessors), 46
localec, ASpliJDU-method
(ASpliJDU-class), 10
localec<-(JDU accessors), 46
localec<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej, 49
localej (JDU accessors), 46
localej, ASpliJDU-method
(ASpliJDU-class), 10
localej<-(JDU accessors), 46
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
localej<-, ASpliJDU, data.frame-method
(ASpliJDU-class), 10
show, ASpliSplicingReport-method (ASpliSplicingReport-class), 11
show-methods, 62
signals, 41
signals (integratedSignals accessors), 39
signals, ASpliIntegratedSignals-method (ASpliIntegratedSignals-class), 10
signals<- (integratedSignals accessors), 39
signals<-, ASpliIntegratedSignals, data.frame-method (ASpliIntegratedSignals-class), 10
splicingReport, 6, 26, 27, 41, 63, 63
splicingReport methods, 64
splicingReport, ASpliDU-method (ASpliDU-class), 9
splicingReport, ASpliIntegratedSignals-method (ASpliIntegratedSignals-class), 10
subset (Subset ASpli objects), 65
Subset ASpli objects, 65
subset, ASpliAS-method (ASpliAS-class), 6
subset, ASpliCounts-method (ASpliCounts-class), 8
subsetBams (Subset ASpli objects), 65
subsetTargets (Subset ASpli objects), 65
targets (Counts accessors), 13
targets, ASpliCounts-method (ASpliCounts-class), 8
targets<-, ASpliCounts, data.frame-method (ASpliCounts-class), 8
tiff, 60
transcriptExons (Features accessors), 28
transcriptExons, ASpliFeatures-method (ASpliFeatures-class), 9
transcriptExons<-, (Features accessors), 28
transcriptExons<-, ASpliFeatures, GRangesList-method (ASpliFeatures-class), 9
write, 66
write-methods, 67
writeAll (write), 66
writeAll, ANY-method (write-methods), 67
writeAll, ASpliCounts-method (ASpliCounts-class), 8
writeAS, 45
writeAS (write), 66
writeAS, ASpliAS-method (ASpliAS-class), 6
writeAS-methods (write-methods), 67
writeCounts, 34
writeCounts (write), 66
writeCounts, ASpliCounts-method (ASpliCounts-class), 8
writeDU, 17, 19, 21, 23, 37, 49, 52
writeDU (write), 66
writeDU, ASpliDU-method (ASpliDU-class), 9
writeDU-methods (write-methods), 67
writeIntegratedSignals, ASpliIntegratedSignals-method (ASpliIntegratedSignals-class), 10
writeJDU, 49
writeJDU (write), 66
writeJDU, ASpliJDU-method (ASpliJDU-class), 10
writeJDU-methods (write-methods), 67
writeRds, write, 66
writeRds, ASpliCounts-method (ASpliCounts-class), 8
writeRds-methods (write-methods), 67
writeSplicingReport, 63
writeSplicingReport (write), 66
writeSplicingReport-methods (write-methods), 67