Package 'SGSeq'

April 23, 2016

Type Package

Title Splice event prediction and quantification from RNA-seq data

Version 1.4.3

Author Leonard Goldstein

Maintainer Leonard Goldstein <goldstel@gene.com>

Description Predict splice junctions and exons from BAM files and obtain compatible read counts and FPKMs. Identify splice events and estimate relative usage of splice variants based on compatible read counts at event boundaries.

License Artistic-2.0

LazyData yes

Depends methods, IRanges, GenomicRanges

Imports AnnotationDbi, BiocGenerics, Biostrings, GenomicAlignments, GenomicFeatures, GenomeInfoDb, igraph, parallel, Rsamtools, rtracklayer, RUnit, S4Vectors, SummarizedExperiment (>= 0.2.0)

Suggests BiocStyle, BSgenome.Hsapiens.UCSC.hg19, knitr, rmarkdown, TxDb.Hsapiens.UCSC.hg19.knownGene

VignetteBuilder knitr

biocViews AlternativeSplicing, RNASeq, Transcription

RoxygenNote 5.0.1

NeedsCompilation no

R topics documented:

`analyzeFeatures` *Analysis of splice graph features from BAM files*

Description

High-level function for the prediction and quantification of splice junctions, exon bins and splice sites from BAM files.

Usage

```
analyzeFeatures(sample_info, which = NULL, features = NULL,
  predict = is.null(features), alpha = 2, psi = 0, beta = 0.2,
  gamma = 0.2, min_n_sample = 1, min_overhang = NA, annotation = NULL,
  max_complexity = 20, verbose = FALSE, cores = 1)
```

Arguments

`sample_info`	Data frame with sample information. Required columns are "sample_name", "file_bam", "paired_end", "read_length", "frag_length" and "lib_size". Library information can be obtained with function `getBamInfo`.
`which`	`GRanges` of genomic regions to be considered for feature prediction, passed to `ScanBamParam`
`features`	`TxFeatures` or `SGFeatures` object

predict	Logical indicating whether transcript features should be predicted from BAM files
alpha	Minimum FPKM required for a splice junction to be included
psi	Minimum splice frequency required for a splice junction to be included
beta	Minimum relative coverage required for an internal exon to be included
gamma	Minimum relative coverage required for a terminal exon to be included
min_n_sample	Minimum number of samples a feature must be observed in to be included
min_overhang	Minimum overhang required to suppress filtering or trimming of predicted terminal exons (see the manual page for processTerminalExons). Use NULL to disable processing (disabling processing is useful if results are subsequently merged with other predictions and processing is postponed until after the merging step).
annotation	TxFeatures object used for annotation
max_complexity	Maximum allowed complexity. If a locus exceeds this threshold, it is skipped, resulting in a warning. Complexity is defined as the maximum number of unique predicted splice junctions overlapping a given position. High complexity regions are often due to spurious read alignments and can slow down processing. To disable this filter, set to NA.
verbose	If TRUE, generate messages indicating progress
cores	Number of cores available for parallel processing

Details

Splice junctions and exons are predicted from BAM files with predictTxFeatures.

Known features can be provided as TxFeatures or SGFeatures via argument features.

If features is not NULL and predict is TRUE, known features are augmented with predictions.

Known and/or predicted transcript features are converted to splice graph features. For details, see convertToSGFeatures.

Optionally, splice graph features can be annotated with respect to a TxFeatures object provided via argument annotation. For details, see the help page for function annotate.

Finally, compatible fragment counts for splice graph features are obtained from BAM files with getSGFeatureCounts.

Value

SGFeatureCounts object

Author(s)

Leonard Goldstein

Examples

```
path <- system.file("extdata", package = "SGSeq")
si$file_bam <- file.path(path, "bams", si$file_bam)
sgfc <- analyzeFeatures(si, gr)
```

analyzeVariants *Analysis of splice variants*

Description

High-level function for the analysis of splice variants from splice graph features. Splice variants are identified with findSGVariants. Representative counts are obtained and variant frequencies estimated with getSGVariantCounts.

Usage

```
analyzeVariants(object, maxnvariant = 20, include = "default",
  min_denominator = NA, cores = 1)
```

Arguments

object	SGFeatureCounts object
maxnvariant	If more than maxnvariant variants are identified in an event, the event is skipped, resulting in a warning. Set to NA to include all events.
include	Character string indicating whether identified splice variants should be filtered. Possible options are "default" (only include variants for events with all variants closed), "closed" (only include closed variants) and "all" (include all variants).
min_denominator	Integer specifying minimum denominator when calculating variant frequencies. If the denominator is smaller than min_denominator, variant frequencies are set to NA. If NA, all variant frequencies are returned.
cores	Number of cores available for parallel processing

Value

An SGVariantCounts object

Author(s)

Leonard Goldstein

Examples

```
sgvc <- analyzeVariants(sgfc_pred)
```

annotate *Annotation with respect to transcript features*

Description

Features in query are annotated with respect to transcript features in subject.

Usage

```
annotate(query, subject)
```

Arguments

query	SGFeatures, SGVariants, SGFeatureCounts or SGVariantCounts object
subject	TxFeatures object

Details

Annotation is performed at the gene and transcript level. For transcript-level annotation, query features are assigned all transcript names associated with matching subject features. For gene-level annotation, query features are assigned all gene names associated with subject features that belong to the same gene (connected component of the splice graph) as matching query features.

Feature matching is performed as follows: Query splice junctions are matched with identical subject splice junctions. Query splice sites are matched with splice sites implied by subject splice junctions. Query exon bins are matched with overlapping subject exons. Spliced boundaries of query exon bins must match spliced subject exon boundaries. Query exon bins cannot extend across spliced subject exon boundaries.

Value

query with updated txName, geneName column slots

Author(s)

Leonard Goldstein

Examples

```
sgf_annotated <- annotate(sgf_pred, txf_ann)
sgv_annotated <- annotate(sgv_pred, txf_ann)
```

assays *Accessing and replacing assay data*

Description

Accessor and replacement functions for assay data.

Usage

```
FPKM(object)

FPKM(object) <- value

countsVariant5p(object)

countsVariant5p(object) <- value

countsVariant3p(object)

countsVariant3p(object) <- value

countsTotal5p(object)

countsTotal5p(object) <- value

countsTotal3p(object)

countsTotal3p(object) <- value

countsVariant(object)

countsVariant(object) <- value

countsTotal(object)

countsTotal(object) <- value

variantFreq(object)

variantFreq(object) <- value

## S4 method for signature 'SGFeatureCounts'
counts(object)

## S4 replacement method for signature 'SGFeatureCounts'
counts(object) <- value
```

```
## S4 method for signature 'SGFeatureCounts'
FPKM(object)

## S4 replacement method for signature 'SGFeatureCounts'
FPKM(object) <- value

## S4 method for signature 'SGVariantCounts'
countsVariant5p(object)

## S4 replacement method for signature 'SGVariantCounts'
countsVariant5p(object) <- value

## S4 method for signature 'SGVariantCounts'
countsVariant3p(object)

## S4 replacement method for signature 'SGVariantCounts'
countsVariant3p(object) <- value

## S4 method for signature 'SGVariantCounts'
countsTotal5p(object)

## S4 replacement method for signature 'SGVariantCounts'
countsTotal5p(object) <- value

## S4 method for signature 'SGVariantCounts'
countsTotal3p(object)

## S4 replacement method for signature 'SGVariantCounts'
countsTotal3p(object) <- value

## S4 method for signature 'SGVariantCounts'
variantFreq(object)

## S4 replacement method for signature 'SGVariantCounts'
variantFreq(object) <- value

## S4 method for signature 'SGVariantCounts'
countsVariant(object)

## S4 replacement method for signature 'SGVariantCounts'
countsVariant(object) <- value

## S4 method for signature 'SGVariantCounts'
countsTotal(object)

## S4 replacement method for signature 'SGVariantCounts'
countsTotal(object) <- value
```

Arguments

object	Object containing assay data
value	Replacement value

Details

Counts objects defined in the SGSeq package contain different types of assay data. For example, class SGFeatureCounts contains assays counts and FPKM.

To facilitate accessing and modifying assays, for each assay there exists a function with name identical to the assay name that can be used to access and modify it (see examples).

Value

Assay data for accessor functions, updated object for replacement functions.

Author(s)

Leonard Goldstein

Examples

```
x <- counts(sgfc_pred)
y <- FPKM(sgfc_pred)
```

---

convertToSGFeatures *Convert transcript features to splice graph features*

Description

Convert transcript features (predicted from RNA-seq data or extracted from transcript annotation) to splice graph features.

Usage

```
convertToSGFeatures(x, coerce = FALSE)
```

Arguments

x	TxFeatures object
coerce	Logical indicating whether transcript features should be coerced to splice graph features without disjoining exons and omitting splice donor and acceptor sites

Details

Splice junctions are unaltered. Exons are disjoined into non-overlapping exon bins. Adjacent exon bins without a splice site at the shared boundary are merged.

Entries for splice donor and acceptor sites (positions immediately upstream and downstream of introns, respectively) are added.

In the returned `SGFeatures` object, column `type` takes values "J" (splice junction), "E" (exon bin), "D" (splice donor) or "A" (splice acceptor). Columns `splice5p` and `splice3p` indicate mandatory splices at the 5' and 3' end of exon bins, respectively (determining whether reads overlapping exon boundaries must be spliced at the boundary to be considered compatible). `splice5p` (`splice3p`) is `TRUE` if the first (last) position of the exon coincides with a splice acceptor (donor) and it is not adjacent to a neighboring exon bin.

Each feature is assigned a unique feature and gene identifier, stored in columns `featureID` and `geneID`, respectively. The latter indicates features that belong to the same gene, represented by a connected component in the splice graph.

Value

An `SGFeatures` object

Author(s)

Leonard Goldstein

Examples

```
sgf <- convertToSGFeatures(txf_ann)
```

---

convertToTxFeatures *Convert to TxFeatures object*

Description

Convert a `TxDb` object or a `GRangesList` of exons grouped by transcripts to a `TxFeatures` object.

Usage

```
convertToTxFeatures(x)
```

Arguments

`x`	`TxDb` object or `GRangesList` of exons grouped by transcript. For import from GFF format, use function `importTranscripts`.

Details

If x is a GRangesList, transcript names and gene names can be specified as character vectors in metadata columns txName and geneName, respectively. If missing, transcript names are based on names(x).

In the returned TxFeatures object, column type takes values "J" (splice junction), "I" (internal exon), "F" (5'/first exon), "L" (3'/last exon) or "U" (unspliced).

Value

A TxFeatures object

Author(s)

Leonard Goldstein

Examples

```
gr <- GRanges(c(1, 1), IRanges(c(1, 201), c(100, 300)), c("+", "+"))
grl <- split(gr, 1)
txf <- convertToTxFeatures(grl)
```

---

exportFeatures *Export to BED format*

Description

Export features to BED format. Splice sites are not included.

Usage

```
exportFeatures(features, file)
```

Arguments

features	TxFeatures or SGFeatures object
file	Character string specifying output file

Value

NULL

Author(s)

Leonard Goldstein

Examples

```
## Not run:
exportFeatures(txf_pred, "txf.bed")
exportFeatures(sgf_pred, "sgf.bed")

## End(Not run)
NULL
```

`findSGVariants` *Identify splice variants from splice graph*

Description

Identify splice variants from splice graph.

Usage

```
findSGVariants(features, maxnvariant = 20, annotate_events = TRUE,
  include = c("default", "closed", "all"), cores = 1)
```

Arguments

`features`	`SGFeatures` object
`maxnvariant`	If more than `maxnvariant` variants are identified in an event, the event is skipped, resulting in a warning. Set to `NA` to include all events.
`annotate_events`	Logical indicating whether identified splice variants should be annotated in terms of canonical events. For details see help page for `annotateSGVariants`.
`include`	Character string indicating whether identified splice variants should be filtered. Possible options are "default" (only include variants for events with all variants closed), "closed" (only include closed variants) and "all" (include all variants).
`cores`	Number of cores available for parallel processing

Value

An `SGVariants` object

Author(s)

Leonard Goldstein

Examples

```
sgv <- findSGVariants(sgf_pred)
```

getBamInfo *Obtain library information from BAM files*

Description

Obtain paired-end status, median aligned read length, median aligned insert size and library size from BAM files.

Usage

```
getBamInfo(sample_info, yieldSize = NULL, cores = 1)
```

Arguments

sample_info	Data frame with sample information including mandatory character columns "sample_name" and "file_bam".
yieldSize	Number of records used for obtaining library information, or NULL for all records
cores	Number of cores available for parallel processing

Details

Library information can be inferred from a subset of BAM records by setting the number of records via argument yieldSize. Note that library size is only obtained if yieldSize is NULL.

Value

sample_info with additional columns "paired_end", "read_length", "frag_length", and "lib_size" if yieldSize is NULL

Author(s)

Leonard Goldstein

Examples

```
path <- system.file("extdata", package = "SGSeq")
si$file_bam <- file.path(path, "bams", si$file_bam)
si <- si[, c("sample_name", "file_bam")]
si_complete <- getBamInfo(si)
```

getSGFeatureCounts *Compatible counts for splice graph features from BAM files*

Description

Compatible counts are obtained for each sample and combined into an SGFeatureCounts object.

Usage

```
getSGFeatureCounts(sample_info, features, counts_only = FALSE,
  verbose = FALSE, cores = 1)
```

Arguments

sample_info	Data frame with sample information. Required columns are "sample_name", "file_bam", "paired_end", "read_length", "frag_length" and "lib_size". Library information can be obtained with function getBamInfo.
features	SGFeatures object
counts_only	Logical indicating only counts should be returned
verbose	If TRUE, generate messages indicating progress
cores	Number of cores available for parallel processing

Value

An SGFeatureCounts object or integer matrix of counts if counts_only = TRUE

Author(s)

Leonard Goldstein

Examples

```
path <- system.file("extdata", package = "SGSeq")
si$file_bam <- file.path(path, "bams", si$file_bam)
sgfc <- getSGFeatureCounts(si, sgf_pred)
```

getSGVariantCounts *Representative counts and frequency estimates for splice variants*

Description

For splice variants obtain counts of compatible fragments extending across the start or end of each variant. Counts can be obtained from an SGFeatureCounts object or from BAM files. Only one of the two arguments object and sample_info must be specified. Splice variant frequencies are estimated based on representive counts.

Usage

```
getSGVariantCounts(variants, object = NULL, features = NULL,
  sample_info = NULL, min_denominator = NA, verbose = FALSE, cores = 1)
```

Arguments

variants	SGVariants object
object	SGFeatureCounts object
features	SGFeatures object that must include all features included in featureID5p(variants) and featureID3p(variants)
sample_info	Data frame with sample information. Required columns are "sample_name", "file_bam", "paired_end", "read_length", "frag_length" and "lib_size". Library information can be obtained with function getBamInfo.
min_denominator	Integer specifying minimum denominator when calculating variant frequencies. If the denominator is smaller than min_denominator, variant frequencies are set to NA. If NA, all variant frequencies are returned.
verbose	If TRUE, generate messages indicating progress
cores	Number of cores available for parallel processing

Value

An SGVariantCounts object

Author(s)

Leonard Goldstein

Examples

```
sgvc_from_sgfc <- getSGVariantCounts(sgv_pred, sgfc_pred)
path <- system.file("extdata", package = "SGSeq")
si$file_bam <- file.path(path, "bams", si$file_bam)
sgvc_from_bam <- getSGVariantCounts(sgv_pred,
  features = sgf_pred, sample_info = si)
```

importTranscripts *Import transcripts from GFF file*

Description

Import GFF file and generate a GRangesList of transcripts suitable as input for functions convertToTxFeatures or predictVariantEffects.

Usage

```
importTranscripts(file, tag_tx = "transcript_id", tag_gene = "gene_id")
```

Arguments

file	Character string specifying input GFF file
tag_tx	GFF attribute tag for transcript identifier
tag_gene	GFF attribute tag for gene identifier

Value

A GRangesList of exons grouped by transcipts with metadata columns txName, geneName, cdsStart, cdsEnd.

Author(s)

Leonard Goldstein

Examples

```
## Not run:
tx <- importTranscripts(file)

## End(Not run)
NULL
```

makeSGFeatureCounts *Create* SGFeatureCounts *object*

Description

Create SGFeatureCounts object from rowRanges, colData and counts.

Usage

```
makeSGFeatureCounts(rowRanges, colData, counts)
```

Arguments

rowRanges	An SGFeatures object
colData	Data frame with sample information
counts	Integer matrix of counts

Value

An SGFeatureCounts object

Author(s)

Leonard Goldstein

Examples

```
sgfc <- makeSGFeatureCounts(sgf_pred, si,
  matrix(0L, length(sgf_pred), nrow(si)))
```

---

mergeTxFeatures *Merge redundant features*

Description

Merge features, typically after feature prediction in multiple samples.

Usage

```
mergeTxFeatures(..., min_n_sample = 1)
```

Arguments

...	one or more TxFeatures objects, or a single list of TxFeatures objects
min_n_sample	Minimum number of samples a feature must be observed in to be included

Details

Merged features are the union of splice junctions and internal exons. For terminal exons with shared spliced boundary, the longest exon is retained.

Value

TxFeatures object with merged features

Author(s)

Leonard Goldstein

Examples

```
txf_merged <- mergeTxFeatures(txf_ann, txf_pred)
```

---

plotCoverage *Plot read coverage and splice junction read counts*

Description

Plot read coverage and splice junction read counts for an individual sample or averaged across samples.

Usage

```
plotCoverage(x, geneID = NULL, geneName = NULL, eventID = NULL,
  which = NULL, sample_info = NULL, sizefactor = NA, toscale = c("exon",
  "none", "gene"), color = "darkblue", ylim = NULL, label = NULL,
  nbin = 200, summary = mean, curvature = 1, main = NULL, cores = 1)
```

Arguments

x	SGFeatureCounts or SGFeatures object. If x is an SGFeatureCounts object that includes multiple samples, average coverage and splice junction counts are obtained.
geneID	Single gene identifier used to subset x
geneName	Single gene name used to subset x
eventID	Single event identifier used to subset x
which	GRanges used to subset x
sample_info	Data frame with sample information. If x is an SGFeatureCounts object, sample information is obtained from colData(x). If sample_info includes multiple samples, average coverage and splice junction counts are obtained.
sizefactor	Numeric vector with length equal to the number of samples in sample_info. Used to scale coverages and splice junction counts before plotting, or before averaging across samples. Set to NA to disable scaling. If NULL, size factors are calculated as the number of bases sequenced (the product of library size and average number of bases sequenced per read or fragment), plotted coverages and splice junction counts are per 1 billion sequenced bases.
toscale	Controls which parts of the splice graph are drawn to scale. Possible values are "none" (exonic and intronic regions have constant length), "exon" (exonic regions are drawn to scale) and "gene" (both exonic and intronic regions are drawn to scale).
color	Color used for plotting coverages
ylim	Numeric vector of length two, determining y-axis range used for plotting coverages.
label	Optional y-axis label

nbin	Number of bins for plotting coverages
summary	Function used to calculate per-bin coverage summaries
curvature	Numeric determining curvature of plotted splice junctions.
main	Plot title
cores	Number of cores available for parallel processing.

Value

data.frame with information on splice junctions included in the splice graph

Author(s)

Leonard Goldstein

Examples

```
## Not run:
par(mfrow = c(4, 1))
for (j in seq_len(4)) plotCoverage(sgfc_pred[, j])

## End(Not run)
NULL
```

---

plotFeatures *Plot splice graph and heatmap of expression values*

Description

Plot splice graph and heatmap of expression values.

Usage

```
plotFeatures(x, geneID = NULL, geneName = NULL, which = NULL,
  tx_view = FALSE, cex = 1, assay = "FPKM", include = c("junctions",
  "exons", "both"), transform = function(x) {     log2(x + 1) },
  Rowv = NULL, distfun = dist, hclustfun = hclust, margin = 0.2,
  RowSideColors = NULL, square = FALSE, cexRow = 1, cexCol = 1,
  labRow = colnames(x), col = colorRampPalette(c("black", "gold"))(256),
  zlim = NULL, heightPanels = c(1, 2), ...)
```

Arguments

x	SGFeatureCounts object
geneID	Single gene identifier used to subset x
geneName	Single gene name used to subset x
which	GRanges used to subset x

tx_view	Plot transcripts instead of splice graph (experimental)
cex	Scale parameter for feature labels and annotation
assay	Name of assay to be plotted in the heatmap
include	Include "exons", "junctions" or "both" in the heatmap
transform	Transformation applied to assay data
Rowv	Determines order of rows. Either a vector of values used to reorder rows, or NA to suppress reordering, or NULL for hierarchical clustering.
distfun	Distance function used for hierarchical clustering of rows (samples)
hclustfun	Clustering function used for hierarchical clustering of rows (samples)
margin	Width of right-hand margin as fraction of width of the graphics device. Ignored if square is TRUE.
RowSideColors	Character vector (or list of character vectors) with length(s) equal to ncol(x) containing color names for horizontal side bars for sample annotation
square	Logical, if TRUE margins are set such that cells in the heatmap are square
cexRow	Scale factor for row (sample) labels
cexCol	Scale factor for column (feature) labels
labRow	Character vector of row (sample) labels
col	Heatmap colors
zlim	Range of values for which colors should be plotted, if NULL range of finite values
heightPanels	Numeric vector of length two indicating height of the top and bottom panels.
...	further parameters passed to plotSpliceGraph

Value

data.frame with information on exon bins and splice junctions included in the splice graph

Author(s)

Leonard Goldstein

Examples

```
## Not run:
sgfc_annotated <- annotate(sgfc_pred, txf_ann)
plotFeatures(sgfc_annotated)

## End(Not run)
NULL
```

plotSpliceGraph *Plot splice graph*

Description

Plot splice graph implied by splice junctions and exon bins.

Usage

```
plotSpliceGraph(x, geneID = NULL, geneName = NULL, eventID = NULL,
  which = NULL, toscale = c("exon", "none", "gene"), label = c("id",
  "name", "label", "none"), color = "gray", color_novel = color,
  color_alpha = 0.8, color_labels = FALSE, border = "fill",
  curvature = NULL, ypos = c(0.5, 0.1), score = NULL,
  score_color = "darkblue", score_ylim = NULL, score_ypos = c(0.3, 0.1),
  score_nbin = 200, score_summary = mean, score_label = NULL,
  ranges = NULL, ranges_color = "darkblue", ranges_ypos = c(0.1, 0.1),
  main = NULL, tx_view = FALSE, tx_dist = 0.2)
```

Arguments

x	SGFeatures or SGVariants object
geneID	Single gene identifier used to subset x
geneName	Single gene name used to subset x
eventID	Single event identifier used to subset x
which	GRanges used to subset x
toscale	Controls which parts of the splice graph are drawn to scale. Possible values are "none" (exonic and intronic regions have constant length), "exon" (exonic regions are drawn to scale) and "gene" (both exonic and intronic regions are drawn to scale).
label	Format of exon/splice junction labels, possible values are "id" (format E1,... J1,...), "name" (format type:chromosome:start-end:strand), "label" for labels specified in metadata column "label", or "none" for no labels.
color	Color used for plotting the splice graph. Ignored if features metadata column "color" is not NULL.
color_novel	Features with missing annotation are highlighted in color_novel. Ignored if features metadata column "color" is not NULL.
color_alpha	Controls color transparency
color_labels	Logical indicating whether label colors should be the same as feature colors
border	Determines the color of exon borders, can be "fill" (same as exon color), "none" (no border) or a valid color name
curvature	Numeric determining curvature of plotted splice junctions.

ypos	Numeric vector of length two, indicating the vertical position and height of the exon bins in the splice graph, specificed as fraction of the height of the plotting region (not supported for `tx_view = TRUE`)
score	`RLeList` containing nucleotide-level scores to be plotted with the splice graph
score_color	Color used for plotting scores
score_ylim	Numeric vector of length two, determining y-axis range for plotting scores
score_ypos	Numeric vector of length two, indicating the vertical position and height of the score panel, specificed as fraction of the height of the plotting region
score_nbin	Number of bins for plotting scores
score_summary	Function used to calculate per-bin score summaries
score_label	Label used to annotate score panel
ranges	`GRangesList` to be plotted with the splice graph
ranges_color	Color used for plotting ranges
ranges_ypos	Numeric vector of length two, indicating the vertical position and height of the ranges panel, specificed as fraction of the height of the plotting region
main	Plot title
tx_view	Plot transcripts instead of splice graph (experimental)
tx_dist	Vertical distance between transcripts as fraction of height of plotting region

Details

By default, the color of features in the splice graph is determined by annotation status (see arguments `color`, `color_novel`) and feature labels are generated automatically (see argument `label`). Alternatively, colors and labels can be specified via metadata columns "color" and "label", respectively.

A `data.frame` with information on plotted features, including genomic coordinates, is returned invisibly.

Value

`data.frame` with information on exon bins and splice junctions included in the splice graph

Author(s)

Leonard Goldstein

Examples

```
## Not run:
sgf_annotated <- annotate(sgf_pred, txf_ann)
plotSpliceGraph(sgf_annotated)

## End(Not run)
## Not run:
sgv_annotated <- annotate(sgv_pred, txf_ann)
plotSpliceGraph(sgv_annotated)
```

```
## End(Not run)
NULL
```

---

plotVariants *Plot splice graph and heatmap of splice variant frequencies*

Description

Plot splice graph and heatmap of splice variant frequencies.

Usage

```
plotVariants(x, eventID = NULL, tx_view = FALSE, cex = 1,
  transform = function(x) {     x }, Rowv = NULL, distfun = dist,
  hclustfun = hclust, margin = 0.2, RowSideColors = NULL,
  square = FALSE, cexRow = 1, cexCol = 1, labRow = colnames(x),
  col = colorRampPalette(c("black", "gold"))(256), zlim = c(0, 1),
  heightPanels = c(1, 2), expand_variants = FALSE, ...)
```

Arguments

x	SGVariantCounts object
eventID	Single event identifier used to subset x
tx_view	Plot transcripts instead of splice graph (experimental)
cex	Scale parameter for feature labels and annotation
transform	Transformation applied to splice variant frequencies
Rowv	Determines order of rows. Either a vector of values used to reorder rows, or NA to suppress reordering, or NULL for hierarchical clustering.
distfun	Distance function used for hierarchical clustering of rows (samples)
hclustfun	Clustering function used for hierarchical clustering of rows (samples)
margin	Width of right-hand margin as fraction of width of the graphics device. Ignored if square is TRUE.
RowSideColors	Character vector (or list of character vectors) with length(s) equal to ncol(x) containing color names for horizontal side bars for sample annotation
square	Logical, if TRUE margins are set such that cells in the heatmap are square
cexRow	Scale factor for row (sample) labels
cexCol	Scale factor for column (feature) labels
labRow	Character vector of row (sample) labels
col	Heatmap colors
zlim	Range of values for which colors should be plotted, if NULL range of finite values
heightPanels	Numeric vector of length two indicating height of the top and bottom panels.
expand_variants	Experimental option - leave set to FALSE
...	further parameters passed to plotSpliceGraph

Value

`data.frame` with information on exon bins and splice junctions included in the splice graph

Author(s)

Leonard Goldstein

Examples

```
## Not run:
sgvc_annotated <- annotate(sgvc_pred, txf_ann)
plotVariants(sgvc_annotated)

## End(Not run)
NULL
```

`predictTxFeatures` *Splice junction and exon prediction from BAM files*

Description

Splice junctions and exons are predicted for each sample and merged across samples. Terminal exons are filtered and trimmed, if applicable. For details, see the help pages for `predictTxFeaturesPerSample`, `mergeTxFeatures`, and `processTerminalExons`.

Usage

```
predictTxFeatures(sample_info, which = NULL, alpha = 2, psi = 0,
  beta = 0.2, gamma = 0.2, min_junction_count = NULL,
  max_complexity = 20, min_n_sample = 1, min_overhang = NA,
  verbose = FALSE, cores = 1)
```

Arguments

`sample_info`	Data frame with sample information. Required columns are "sample_name", "file_bam", "paired_end", "read_length", "frag_length" and "lib_size". Library information can be obtained with function `getBamInfo`.
`which`	`GRanges` of genomic regions to be considered for feature prediction, passed to `ScanBamParam`
`alpha`	Minimum FPKM required for a splice junction to be included. Internally, FPKMs are converted to counts, requiring arguments `read_length`, `frag_length` and `lib_size`. `alpha` is ignored if argument `min_junction_count` is specified.
`psi`	Minimum splice frequency required for a splice junction to be included
`beta`	Minimum relative coverage required for an internal exon to be included
`gamma`	Minimum relative coverage required for a terminal exon to be included

`min_junction_count`
: Minimum fragment count required for a splice junction to be included. If specified, argument `alpha` is ignored.

`max_complexity`
: Maximum allowed complexity. If a locus exceeds this threshold, it is skipped, resulting in a warning. Complexity is defined as the maximum number of unique predicted splice junctions overlapping a given position. High complexity regions are often due to spurious read alignments and can slow down processing. To disable this filter, set to `NA`.

`min_n_sample`
: Minimum number of samples a feature must be observed in to be included

`min_overhang`
: Minimum overhang required to suppress filtering or trimming of predicted terminal exons (see the manual page for `processTerminalExons`). Use `NULL` to disable processing (disabling processing is useful if results are subsequently merged with other predictions and processing is postponed until after the merging step).

`verbose`
: If `TRUE`, generate messages indicating progress

`cores`
: Number of cores available for parallel processing

Value

A `TxFeatures` object

Author(s)

Leonard Goldstein

Examples

```
path <- system.file("extdata", package = "SGSeq")
si$file_bam <- file.path(path, "bams", si$file_bam)
txf <- predictTxFeatures(si, gr)
```

---

predictVariantEffects *Predict the effect of splice variants on protein-coding transcripts*

Description

The effect of each splice variant is assessed with respect to individual protein-coding transcripts.

Usage

```
predictVariantEffects(sgv, tx, genome, summarize = TRUE, cores = 1)
```

Arguments

sgv	SGVariants object
tx	A TxDb object or GRangesList of exons grouped by transcript with metadata columns cdsStart and cdsEnd (by convention, cdsStart < cdsEnd for both strands). For import from GFF format, use function importTranscripts.
genome	BSgenome object
summarize	Logical indicating whether results should be summarized per variant
cores	Number of cores available for parallel processing

Value

For summarize = FALSE a data.frame with rows corresponding to a variant-transcript pair. The data.frame includes columns for variant identifier, transcript name, type of alteration, protein sequences for the reference transcript and the transcript variant, protein lengths and coordinates of the variant in the protein sequences. Start and end coordinates are 0- and 1-based, respectively, to allow for specification of deletions. For summarize = TRUE a character vector matching argument sgv with comma-separated predicted alterations for individual transcripts.

Author(s)

Leonard Goldstein

Examples

```
require(BSgenome.Hsapiens.UCSC.hg19)
seqlevelsStyle(Hsapiens) <- "NCBI"
predictVariantEffects(sgv_pred, tx, Hsapiens)
```

---

processTerminalExons *Process predicted terminal exons*

Description

Predicted terminal exons are processed as described under Details.

Usage

```
processTerminalExons(features, min_overhang = NA)
```

Arguments

features	TxFeatures object
min_overhang	Minimum overhang required to suppress filtering or trimming of predicted terminal exons (see Details). Use NA to exclude all terminal exons sharing a splice with an internal exon and trim all remaining terminal exons overlapping other exons.

Details

Processing of terminal exon predictions is done in two steps: (1) terminal exons that share a splice site with an internal exon are filtered, and (2) remaining terminal exons that overlap other exons are trimmed.

`predictTxFeatures` predicts flanking terminal exons for each identified splice junction. This ensures that each splice junction has a flanking exon after merging with `mergeTxFeatures`. This approach results in many predicted terminal exons that share a splice site with predicted internal exons (often contained within them or with a short overhang due to incorrect alignments). Most of these are not real terminal exons and are filtered before further analysis. Filtering based on the overhang is controlled with argument `min_overhang`.

Some of the remaining predicted terminal exons overlap other exons such that their unspliced boundary shows a short overhang with respect to a spliced boundary of the overlapping exon. Often these exon extensions into an intron are due to incorrect alignments. Terminal exons with overhang smaller than `min_overhang` are trimmed such that their trimmmed unspliced boundary coincides with the spliced boundary of the overlapping exon.

Value

`TxFeatures` object with processed features

Author(s)

Leonard Goldstein

Examples

```
txf_processed <- processTerminalExons(txf_ann)
```

SGFeatureCounts *Constructor function for S4 class* SGFeatureCounts

---

Description

Creates an instance of S4 class `SGFeatureCounts` for storing compatible splice graph feature counts.

Usage

```
SGFeatureCounts(x)
```

Arguments

`x`	`RangedSummarizedExperiment` with `SGFeatures` as `rowRanges` and assays "counts", "FPKM"

Value

An SGFeatureCounts object

Author(s)

Leonard Goldstein

Examples

```
sgfc <- SGFeatureCounts()
```

SGFeatures *Constructor function for S4 class* SGFeatures

Description

Creates an instance of S4 class SGFeatures for storing splice graph features.

Usage

```
SGFeatures(x, type = mcols(x)$type, splice5p = mcols(x)$splice5p,
  splice3p = mcols(x)$splice3p, featureID = mcols(x)$featureID,
  geneID = mcols(x)$geneID, txName = mcols(x)$txName,
  geneName = mcols(x)$geneName)
```

Arguments

x	GRanges with known strand ("+", "-")
type	Character vector or factor taking values in J, E, D, A
splice5p	Logical vector indicating a mandatory splice at the 5' end of an exon bin (determining whether reads extending across the 5' boundary must be spliced to be considered compatible)
splice3p	Logical vector indicating a mandatory splice at the 3' end of an exon bin (determining whether reads extending across the 3' boundary must be spliced to be considered compatible)
featureID	Integer vector of feature IDs
geneID	Integer vector of gene IDs
txName	CharacterList of transcript names or NULL
geneName	CharacterList of gene names or NULL

Details

SGFeatures extends GRanges with column slot type specifying feature type. type is a factor with levels J (splice junction), E (exon bin), D (splice donor), A (splice acceptor).

splice5p and splice3p are logical vectors indicating mandatory splices at the 5' and 3' end of an exon bin, respectively. These are used to determine whether reads extending across the 5' and 3' boundaries of an exon bin must be spliced at the boundary to be considered compatible with the exon bin.

featureID and geneID are integer vectors representing unique identifiers for features and genes (connected components in the splice graph).

txName and geneName are CharacterLists storing transcript and gene annotation, respectively.

Value

An SGFeatures object

Author(s)

Leonard Goldstein

Examples

```
sgf <- SGFeatures()
```

SGVariantCounts *Constructor function for S4 class* SGFeatureCounts

---

Description

Creates an instance of S4 class SGVariantCounts for storing representative splice variant counts.

Usage

```
SGVariantCounts(x)
```

Arguments

x	RangedSummarizedExperiment with SGVariants as rowRanges and appropriate assays

Value

A SGVariantCounts object

Author(s)

Leonard Goldstein

Examples

```
sgvc <- SGVariantCounts()
```

SGVariants *Constructor function for S4 class* SGVariants

Description

Creates an instance of S4 class SGVariants for storing splice variants.

Usage

```
SGVariants(x)
```

Arguments

x	GRangesList of SGFeatures with appropriate outer metadata columns

Value

A SGVariants object

Author(s)

Leonard Goldstein

Examples

```
sgv <- SGVariants()
```

---

slots *Accessing and replacing metadata columns*

Description

Accessor and replacement functions for metadata columns.

Usage

```
type(object)

type(object) <- value

txName(object)

txName(object) <- value

geneName(object)

geneName(object) <- value

featureID(object)

featureID(object) <- value

geneID(object)

geneID(object) <- value

splice5p(object)

splice5p(object) <- value

splice3p(object)

splice3p(object) <- value

from(object)

from(object) <- value

to(object)

to(object) <- value

segmentID(object)

segmentID(object) <- value

variantID(object)

variantID(object) <- value

eventID(object)

eventID(object) <- value
```

```
closed5p(object)

closed5p(object) <- value

closed3p(object)

closed3p(object) <- value

variantType(object)

variantType(object) <- value

variantName(object)

variantName(object) <- value

featureID5p(object)

featureID5p(object) <- value

featureID3p(object)

featureID3p(object) <- value

## S4 method for signature 'Features'
type(object)

## S4 method for signature 'Paths'
type(object)

## S4 method for signature 'Counts'
type(object)

## S4 replacement method for signature 'Features'
type(object) <- value

## S4 replacement method for signature 'Paths'
type(object) <- value

## S4 replacement method for signature 'Counts'
type(object) <- value

## S4 method for signature 'Features'
txName(object)

## S4 method for signature 'Paths'
txName(object)
```

```
## S4 method for signature 'Counts'
txName(object)

## S4 replacement method for signature 'Features'
txName(object) <- value

## S4 replacement method for signature 'Paths'
txName(object) <- value

## S4 replacement method for signature 'Counts'
txName(object) <- value

## S4 method for signature 'Features'
geneName(object)

## S4 method for signature 'Paths'
geneName(object)

## S4 method for signature 'Counts'
geneName(object)

## S4 replacement method for signature 'Features'
geneName(object) <- value

## S4 replacement method for signature 'Paths'
geneName(object) <- value

## S4 replacement method for signature 'Counts'
geneName(object) <- value

## S4 method for signature 'SGFeatures'
featureID(object)

## S4 method for signature 'Paths'
featureID(object)

## S4 method for signature 'Counts'
featureID(object)

## S4 replacement method for signature 'SGFeatures'
featureID(object) <- value

## S4 replacement method for signature 'Paths'
featureID(object) <- value

## S4 replacement method for signature 'Counts'
featureID(object) <- value
```

```
## S4 method for signature 'SGFeatures'
geneID(object)

## S4 method for signature 'Paths'
geneID(object)

## S4 method for signature 'Counts'
geneID(object)

## S4 replacement method for signature 'SGFeatures'
geneID(object) <- value

## S4 replacement method for signature 'Paths'
geneID(object) <- value

## S4 replacement method for signature 'Counts'
geneID(object) <- value

## S4 method for signature 'SGFeatures'
splice5p(object)

## S4 method for signature 'SGSegments'
splice5p(object)

## S4 method for signature 'SGFeatureCounts'
splice5p(object)

## S4 replacement method for signature 'SGFeatures'
splice5p(object) <- value

## S4 replacement method for signature 'SGSegments'
splice5p(object) <- value

## S4 replacement method for signature 'SGFeatureCounts'
splice5p(object) <- value

## S4 method for signature 'SGFeatures'
splice3p(object)

## S4 method for signature 'SGSegments'
splice3p(object)

## S4 method for signature 'SGFeatureCounts'
splice3p(object)

## S4 replacement method for signature 'SGFeatures'
splice3p(object) <- value
```

```
## S4 replacement method for signature 'SGSegments'
splice3p(object) <- value

## S4 replacement method for signature 'SGFeatureCounts'
splice3p(object) <- value

## S4 method for signature 'Paths'
segmentID(object)

## S4 method for signature 'SGVariantCounts'
segmentID(object)

## S4 replacement method for signature 'Paths'
segmentID(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
segmentID(object) <- value

## S4 method for signature 'Paths'
from(object)

## S4 method for signature 'SGVariantCounts'
from(object)

## S4 replacement method for signature 'Paths'
from(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
from(object) <- value

## S4 method for signature 'Paths'
to(object)

## S4 method for signature 'SGVariantCounts'
to(object)

## S4 replacement method for signature 'Paths'
to(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
to(object) <- value

## S4 method for signature 'SGVariants'
eventID(object)

## S4 method for signature 'SGVariantCounts'
eventID(object)
```

```
## S4 replacement method for signature 'SGVariants'
eventID(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
eventID(object) <- value

## S4 method for signature 'SGVariants'
variantID(object)

## S4 method for signature 'SGVariantCounts'
variantID(object)

## S4 replacement method for signature 'SGVariants'
variantID(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
variantID(object) <- value

## S4 method for signature 'SGVariants'
closed5p(object)

## S4 method for signature 'SGVariantCounts'
closed5p(object)

## S4 replacement method for signature 'SGVariants'
closed5p(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
closed5p(object) <- value

## S4 method for signature 'SGVariants'
closed3p(object)

## S4 method for signature 'SGVariantCounts'
closed3p(object)

## S4 replacement method for signature 'SGVariants'
closed3p(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
closed3p(object) <- value

## S4 method for signature 'SGVariants'
variantName(object)

## S4 method for signature 'SGVariantCounts'
variantName(object)
```

```
## S4 replacement method for signature 'SGVariants'
variantName(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
variantName(object) <- value

## S4 method for signature 'SGVariants'
variantType(object)

## S4 method for signature 'SGVariantCounts'
variantType(object)

## S4 replacement method for signature 'SGVariants'
variantType(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
variantType(object) <- value

## S4 method for signature 'SGVariants'
featureID5p(object)

## S4 method for signature 'SGVariantCounts'
featureID5p(object)

## S4 replacement method for signature 'SGVariants'
featureID5p(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
featureID5p(object) <- value

## S4 method for signature 'SGVariants'
featureID3p(object)

## S4 method for signature 'SGVariantCounts'
featureID3p(object)

## S4 replacement method for signature 'SGVariants'
featureID3p(object) <- value

## S4 replacement method for signature 'SGVariantCounts'
featureID3p(object) <- value
```

Arguments

object	Object containing metadata column
value	Replacement value

Details

S4 classes defined in the SGSeq package contain metadata columns that store information for each element in the object. For example, class TxFeatures contains a column type that indicates feature type. The specific columns contained in an object depend on its class.

To facilitate accessing and modifying metadata columns, for each column there exists a function with name identical to the column name that can be used to access and modify it (see examples).

Value

Content of metadata column for accessor functions or updated object for replacement functions.

Author(s)

Leonard Goldstein

Examples

```
head(type(txf_ann))
head(type(sgf_ann))
```

TxFeatures *Constructor function for S4 class* TxFeatures

Description

Creates an instance of S4 class TxFeatures for storing transcript features.

Usage

```
TxFeatures(x, type = mcols(x)$type, txName = mcols(x)$txName,
  geneName = mcols(x)$geneName)
```

Arguments

x	GRanges with known strand ("+", "-")
type	Character vector or factor, taking values in J, I, F, L, U
txName	CharacterList of transcript names or NULL
geneName	CharacterList of gene names or NULL

Details

TxFeatures extends GRanges with column slot type specifying feature type. type is a factor with levels J (splice junction), I (internal exon), F (5' terminal exon), L (3' terminal exon), U (unspliced transcript).

txName and geneName are CharacterLists storing transcript and gene annotation, respectively.

Value

A TxFeatures object

Author(s)

Leonard Goldstein

Examples

```
gr <- GRanges(1, IRanges(101, 200), "+")
txf <- TxFeatures(gr, type = "J")
```

Index