Package ‘SGSeq’

April 25, 2024

Type Package

Title Splice event prediction and quantification from RNA-seq data

Version 1.37.0

Description SGSeq is a software package for analyzing splice events from RNA-seq data. Input data are RNA-seq reads mapped to a reference genome in BAM format. Genes are represented as a splice graph, which can be obtained from existing annotation or predicted from the mapped sequence reads. Splice events are identified from the graph and are quantified locally using structurally compatible reads at the start or end of each splice variant. The software includes functions for splice event prediction, quantification, visualization and interpretation.

License Artistic-2.0

LazyData yes

Depends R (>= 4.0), IRanges (>= 2.13.15), GenomicRanges (>= 1.31.10), Rsamtools (>= 1.31.2), SummarizedExperiment, methods

Imports AnnotationDbi, BiocGenerics (>= 0.31.5), Biostrings (>= 2.47.6), GenomicAlignments (>= 1.15.7), GenomicFeatures (>= 1.31.5), GenomeInfoDb, RUnit, S4Vectors (>= 0.23.19), grDevices, graphics, igraph, parallel, rtracklayer (>= 1.39.7), stats

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

VignetteBuilder knitr

biocViews AlternativeSplicing, ImmunoOncology, RNASeq, Transcription

RoxygenNote 6.0.1

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

git_branch devel

git_last_commit 756479a

git_last_commit_date 2023-10-24

Repository Bioconductor 3.19

Date/Publication 2024-04-24

Author Leonard Goldstein [cre, aut]

Maintainer Leonard Goldstein <ldgoldstein@gmail.com>

Contents

analyzeFeatures . . . 3
analyzeVariants . . . 5
annotate . . . 6
annotateSGVariants . . . 7
assays . . . 8
convertToSGFeatures . . . 9
convertToTxFeatures . . . 10
exonCompatible . . . 11
exportFeatures . . . 12
filterFeatures . . . 13
findOverlapsRanges . . . 14
findSGVariants . . . 14
getBamInfo . . . 15
getSGFeatureCounts . . . 16
getSGFeatureCountsPerSample . . . 17
getSGVariantCounts . . . 18
gr . . . 19
importTranscripts . . . 19
junctionCompatible . . . 20
makeSGFeatureCounts . . . 21
makeVariantNames . . . 21
mergeTxFeatures . . . 22
plotCoverage . . . 23
plotFeatures . . . 24
plotSpliceGraph . . . 26
plotVariants . . . 28
predictCandidatesInternal . . . 29
predictCandidatesTerminal . . . 30
predictExonsInternal . . . 30
predictExonsTerminal . . . 31
predictJunctions . . . 32
predictSpliced . . . 33
predictTxFeatures . . . 34
predictTxFeaturesPerSample . . . 35
predictTxFeaturesPerStrand . . . 37
predictVariantEffects . . . 38
processTerminalExons . . . 39
removeExonsIsolated . . . 40
sgfc_ann . . . 41
sgfc_pred . . . 41
SGFeatureCounts . . . 41

SGFeatures 42
sgf_ann 43
sgf_pred 44
SGSegments 44
SGVariantCounts 45
SGVariants 45
sgvc_ann 47
sgvc_ann_from_bam 47
sgvc_pred 47
sgvc_pred_from_bam 48
sgv_ann 48
sgv_pred 48
si 49
slots 49
splicesiteOverlap 58
tx 59
TxFeatures 59
txf_ann 60
txf_pred 60
updateObject 61

Index **62**

`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_junction_count = NULL, min_anchor = 1,
  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_junction_count	Minimum fragment count required for a splice junction to be included. If specified, argument alpha is ignored.
min_anchor	Integer specifiying minimum anchor length
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, min_anchor = 1, 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. The total number of boundary-spanning reads must be equal to or greater than `min_denominator` for at least one event boundary. Otherwise estimates are set to `NA`. If `NA`, all estimates are returned.
`min_anchor`	Integer specifiying minimum anchor length
`cores`	Number of cores available for parallel processing

Value

`SGVariantCounts` object

Author(s)

Leonard Goldstein

Examples

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

annotate *Annotation with respect to transcript features*

Description

Features in `query` are assigned transcript names and gene names of structurally compatible features in `subject` (see below). If a feature in `query` does not match any features in `subject`, its geneName inherits from connected annotated features.

Usage

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

Arguments

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

Details

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)
```

`annotateSGVariants` *Annotate splice variants in terms of canonical events*

Description

Annotate splice variants in terms of canonical events.

Usage

```
annotateSGVariants(variants)
```

Arguments

`variants`	`SGVariants` object

Details

The following events are considered:

"SE" skipped exon

"S2E" two consecutive exons skipped

"RI" retained intron

"MXE" mutually exclusive exons

"A5SS" alternative 5' splice site

"A3SS" alternative 3' splice site

"AFE" alternative first exon

"ALE" alternative last exon

"AS" alternative start other than "AFE"

"AE" alternative end other than "ALE"

For events "SE" and "S2E", suffixes "I" and "S" indicate inclusion and skipping, respectively. For event "RI" suffixes "E" and "R" indicate exclusion and retention, respectively. For events "A5SS" and "A3SS", suffixes "P" and "D" indicate use of the proximal (intron-shortening) and distal (intron-lengthening) splice site, respectively.

All considered events are binary events defined by two alternative variants. A variant is annotated as a canonical event if it coincides with one of the two variants in the canonical event, and there is at least one variant in the same event that coincides with the second variant of the canonical event.

Value

`variants` with added metadata column "variantType" indicating canonical event(s)

Author(s)

Leonard Goldstein

assays *Accessing and replacing assay data*

Description

Functions counts and FPKM are used to extract counts and FPKM values from SGFeatureCounts and SGVariantCounts objects. Function variantFreq is used to access relative usage estimates from SGVariantCounts objects.

Usage

```
FPKM(object, ...)

FPKM(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'
counts(object, ...)

## S4 replacement method for signature 'SGVariantCounts'
counts(object, ...) <- value

## S4 method for signature 'SGVariantCounts'
FPKM(object, ...)

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

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

Arguments

`object`	Object containing assay data
`...`	Arguments passed to method for `SGVariantCounts` objects. Argument `option` specifies whether the output should be based on the count of fragments compatible with the variant at the start ("variant5p"), end ("variant3p") or either ("variant5pOr3p") (the default), or whether output should be based on the count of fragments compatible with any variant belonging to the same event ("event5p" or "event3p"). Argument `min_anchor` specifies the minimum anchor length when computing FPKM values (defaults to 1).
`value`	Replacement value

Value

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

Author(s)

Leonard Goldstein

Examples

```
x <- counts(sgfc_pred)
y <- FPKM(sgfc_pred)
u <- counts(sgvc_pred, option = "variant5p")
v <- FPKM(sgvc_pred, option = "variant5p")
```

`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

`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). For import from GFF format, use function importTranscripts.

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

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)
```

---

exonCompatible *Compatible fragment counts for exons*

Description

Identify fragments compatible with exons.

Usage

```
exonCompatible(exons, spliceL, spliceR, frag_exonic, frag_intron,
  counts = TRUE)
```

Arguments

exons	IRanges of exons
spliceL	Logical vector indicating whether LHS boundary is spliced
spliceR	Logical vector indicating whether RHS boundary is spliced
frag_exonic	IRangesList of exonic regions, one entry per fragment
frag_intron	IRangesList of introns, one entry per fragment
counts	Logical indicating whether counts or indices of compatible fragments should be returned

Value

Counts or list of indices of compatible fragments

Author(s)

Leonard Goldstein

`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
```

filterFeatures *Filter predicted features*

Description

Filter previously predicted features using more stringent criteria.

Usage

```
filterFeatures(features, paired_end, read_length, frag_length, lib_size,
  min_junction_count = NULL, alpha, psi, beta, gamma)
```

Arguments

features	TxFeatures object with predicted features, including metadata columns "N", "N_splicesite" and "coverage".
paired_end	Logical, TRUE for paired-end data, FALSE for single-end data
read_length	Read length required for use with alpha
frag_length	Fragment length for paired-end data required for use with alpha
lib_size	Number of aligned fragments required for use with alpha
min_junction_count	Minimum fragment count required for a splice junction to be included. If specified, argument alpha is ignored.
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

Details

Initial predictions with predictTxFeatures must have been performed with include_counts = TRUE and retain_coverage = TRUE, so that predicted features contain metadata columns "N", "N_splicesite" and "coverage".

Value

TxFeatures object with filtered features

Author(s)

Leonard Goldstein

findOverlapsRanges *Modified* findOverlaps *function for* IRanges, IRangesList *objects*

Description

Modified findOverlaps function for IRanges, IRangesList objects that behaves analogous to findOverlaps for GRanges, GRangesList objects.

Usage

```
findOverlapsRanges(query, subject, type = "any")
```

Arguments

query	IRanges or IRangesList object
subject	IRanges or IRangesList object
type	Passed to findOverlaps

Value

Hits object

Author(s)

Leonard Goldstein

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

`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 columns "sample_name" and "file_bam". Column "sample_name" must be a character vector. Column "file_bam" can be a character vector or `BamFileList`.
`yieldSize`	Number of records used for obtaining library information, or `NULL` for all records
`cores`	Number of cores available for parallel processing

Details

BAM files must have been generated with a splice-aware alignment program that outputs the custom tag 'XS' for spliced reads, indicating the direction of transcription. BAM files must be indexed.

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)

## data.frame as sample_info and character vector as file_bam
si <- si[, c("sample_name", "file_bam")]
si_complete <- getBamInfo(si)

## DataFrame as sample_info and BamFileList as file_bam
DF <- DataFrame(si)
DF$file_bam <- BamFileList(DF$file_bam)
DF_complete <- getBamInfo(DF)
```

`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, min_anchor = 1,
  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

min_anchor	Integer specifiying minimum anchor length
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

codeSGFeatureCounts 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)
```

---

getSGFeatureCountsPerSample

Compatible fragment counts for splice graph features

Description

Obtain counts of compatible fragments for splice graph features.

Usage

```
getSGFeatureCountsPerSample(features, file_bam, paired_end, sample_name,
  min_anchor, retain_coverage, verbose, cores)
```

Arguments

features	SGFeatures object
file_bam	BAM file with genomic RNA-seq read alignments
paired_end	Logical, TRUE for paired-end data, FALSE for single-end data
sample_name	Sample name used in messages
min_anchor	Integer specifiying minimum anchor length
retain_coverage	Logical indicating whether coverage for each exon should be retained as an RleList in metadata column "coverage". This allows filtering of features using more stringent criteria after the initial prediction.
verbose	If TRUE, generate messages indicating progress
cores	Number of cores available for parallel processing

Value

Numeric vector of compatible fragment counts

Author(s)

Leonard Goldstein

---

getSGVariantCounts *Representative counts and frequency estimates for splice variants*

Description

For splice variants, obtain counts of compatible fragments spanning the start and/or end of each variant. Counts can be obtained from an `SGFeatureCounts` object or from BAM files. Only one of the two arguments `feature_counts` or `sample_info` must be specified. Local estimates of relative usage are calculated at the start and/or end of each splice variant. For splice variants with relative usage estimates at both start and end, these are combined by taking a weighted mean, where weights are proportional to the total number of reads spanning the respective boundary.

Usage

```
getSGVariantCounts(variants, feature_counts = NULL, sample_info = NULL,
  min_denominator = NA, min_anchor = 1, verbose = FALSE, cores = 1)
```

Arguments

`variants`	`SGVariants` object
`feature_counts`	`SGFeatureCounts` object
`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. The total number of boundary-spanning reads must be equal to or greater than `min_denominator` for at least one event boundary. Otherwise estimates are set to `NA`. If `NA`, all estimates are returned.
`min_anchor`	Integer specifiying minimum anchor length
`verbose`	If `TRUE`, generate messages indicating progress
`cores`	Number of cores available for parallel processing

Value

`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, sample_info = si)
```

---

gr — *Example genomic region of interest*

Description

FBXO31 gene locus, based on UCSC knownGene annotation.

Format

`GRanges` object

Author(s)

Leonard Goldstein

---

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

`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
```

---

junctionCompatible *Compatible fragment counts for splice junctions*

Description

Identify fragments compatible with splice junctions.

Usage

```
junctionCompatible(junctions, frag_exonic, frag_intron, min_anchor,
  counts = TRUE)
```

Arguments

`junctions`	`IRanges` of splice junctions
`frag_exonic`	`IRangesList` of exonic regions, one entry per fragment
`frag_intron`	`IRangesList` of introns, one entry per fragment
`min_anchor`	Integer specifying minimum anchor length
`counts`	Logical indicating whether counts or indices of compatible fragments should be returned

Value

Counts or list of indices of compatible fragments

Author(s)

Leonard Goldstein

makeSGFeatureCounts *Create* SGFeatureCounts *object*

Description

Create SGFeatureCounts object from rowRanges, colData and counts.

Usage

```
makeSGFeatureCounts(rowRanges, colData, counts, min_anchor = 1)
```

Arguments

rowRanges	SGFeatures object
colData	Data frame with sample information
counts	Integer matrix of counts
min_anchor	Integer specifiying minimum anchor length

Value

SGFeatureCounts object

Author(s)

Leonard Goldstein

Examples

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

makeVariantNames *Create interpretable splice variant names*

Description

Create interpretable splice variant names taking format GENE_EVENT_VARIANT/ORDER_TYPE. GENE is based on geneName if available, and geneID otherwise. EVENT and VARIANT enumerate events and variants for the same gene and event, respectively. ORDER indicates the total number of variants in the same event (e.g. 1/2 refers to the first out of two splice variants in the event). TYPE is based on variantType.

Usage

```
makeVariantNames(variants)
```

Arguments

`variants`	`SGVariants` object

Value

Character vector with splice variant names

Author(s)

Leonard Goldstein

Examples

```
makeVariantNames(sgv_pred)
```

---

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,
  min_anchor = 1, 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
min_anchor	Integer specifiying minimum anchor length
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 arguments 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 the splice graph implied by splice junctions and exon bins. Invisibly returns a `data.frame` with details of plotted features, including genomic coordinates.

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, short_output = TRUE)
```

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
short_output	Logical indicating whether the returned data frame should only include information that is likely useful to the user

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.

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 arguments 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
```

---

predictCandidatesInternal

Identify candidate internal exons

Description

Identify candidate internal exons based on previously identified splice sites and regions with sufficient read coverage.

Usage

```
predictCandidatesInternal(islands, splicesites, frag_coverage, relCov)
```

Arguments

`islands`	`IRanges` of genomic regions with minimal read coverage required for internal exon prediction
`splicesites`	`IRanges` of splice sites with metadata columns "type" and "N"
`frag_coverage`	`Rle` object with fragment coverage
`relCov`	Minimum relative coverage required for exon prediction

Value

`IRanges` of candidate internal exons

Author(s)

Leonard Goldstein

predictCandidatesTerminal *Identify candidate terminal exons*

Description

Identify candidate terminal exons based on previously identified splice sites and regions with sufficient read coverage.

Usage

```
predictCandidatesTerminal(islands, splicesites, type = c("exon_L", "exon_R"))
```

Arguments

islands	IRanges of genomic regions with minimal read coverage required for internal exon prediction
splicesites	IRanges of splice sites with metadata columns "type" and "N"
type	Character string indicating whether terminal exons should be identified to the left ("exon_L") or right ("exon_R") of provided splice sites

Value

IRanges of candidate terminal exons

Author(s)

Leonard Goldstein

predictExonsInternal *Identify internal exons*

Description

Identify internal exons based on candidate internal exons and compatible read coverage.

Usage

```
predictExonsInternal(candidates, frag_exonic, frag_intron, relCov, min_anchor,
  include_counts, retain_coverage)
```

Arguments

`candidates`	`IRanges` of candidate internal exons
`frag_exonic`	`IRangesList` with exonic regions from alignments
`frag_intron`	`IRangesList` with introns implied by spliced alignments
`relCov`	Minimum relative coverage required for exon prediction
`min_anchor`	Integer specifiying minimum anchor length
`include_counts`	Logical indicating whether counts of compatible fragments should be included in metadata column "N"
`retain_coverage`	Logical indicating whether coverage for each exon should be retained as an `RleList` in metadata column "coverage". This allows filtering of features using more stringent criteria after the initial prediction.

Value

`IRanges` of internal exons with metadata column "type" and optionally "N" for `include_counts = TRUE`, "N_splicesite", "coverage" for `retain_coverage = TRUE`

Author(s)

Leonard Goldstein

`predictExonsTerminal` *Identify terminal exons*

Description

Identify terminal exons based on candidate terminal exons and compatible read coverage.

Usage

```
predictExonsTerminal(candidates, frag_exonic, frag_intron, relCov, min_anchor,
  type = c("exon_L", "exon_R"), include_counts, retain_coverage)
```

Arguments

`candidates`	`IRanges` of candidate internal exons
`frag_exonic`	`IRangesList` with exonic regions from alignments
`frag_intron`	`IRangesList` with introns implied by spliced alignments
`relCov`	Minimum relative coverage required for exon prediction
`min_anchor`	Integer specifiying minimum anchor length
`type`	Character string indicating whether terminal exons should be identified to the left ("exon_L") or right ("exon_R") of provided splice sites

`include_counts` Logical indicating whether counts of compatible fragments should be included in metadata column "N"

`retain_coverage` Logical indicating whether coverage for each exon should be retained as an `RleList` in metadata column "coverage". This allows filtering of features using more stringent criteria after the initial prediction.

Value

`IRanges` of terminal exons with metadata column "type" and optionally "N" for `include_counts = TRUE`, "N_splicesite", "coverage" for `retain_coverage = TRUE`

Author(s)

Leonard Goldstein

`predictJunctions` *Identify splice junctions*

Description

Identify splice junctions from genomic RNA-seq read alignments.

Usage

```
predictJunctions(frag_exonic, frag_intron, min_junction_count, psi, min_anchor,
  retain_coverage)
```

Arguments

`frag_exonic` `IRangesList` with exonic regions from alignments

`frag_intron` `IRangesList` with introns implied by spliced alignments

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

`psi` Minimum splice frequency required for a splice junction to be included

`min_anchor` Integer specifiying minimum anchor length

`retain_coverage` Logical indicating whether coverage for each exon should be retained as an `RleList` in metadata column "coverage". This allows filtering of features using more stringent criteria after the initial prediction.

Value

`IRanges` of splice junctions with metadata columns "type" and "N", and optionally "N_splicesite" for `retain_coverage = TRUE`

Author(s)

Leonard Goldstein

`predictSpliced` *Ranges-based identification of splice junctions and exons*

Description

Ranges-based identification of splice junctions and exons.

Usage

```
predictSpliced(frag_exonic, frag_intron, min_junction_count, psi, beta, gamma,
  min_anchor, include_counts, retain_coverage, junctions_only, max_complexity,
  sample_name, seqlevel, strand)
```

Arguments

`frag_exonic`	`IRangesList` with exonic regions from alignments
`frag_intron`	`IRangesList` with introns implied by spliced alignments
`min_junction_count`	Minimum fragment count required for a splice junction to be included. If specified, argument `alpha` is ignored.
`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_anchor`	Integer specifiying minimum anchor length
`include_counts`	Logical indicating whether counts of compatible fragments should be included in metadata column "N"
`retain_coverage`	Logical indicating whether coverage for each exon should be retained as an `RleList` in metadata column "coverage". This allows filtering of features using more stringent criteria after the initial prediction.
`junctions_only`	Logical indicating whether predictions should be limited to identification of splice junctions only
`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`.
`sample_name`	Sample name used in messages
`seqlevel`	`seqlevel` to be processed
`strand`	`strand` to be processed

Value

IRanges with predicted features

Author(s)

Leonard Goldstein

---

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, min_anchor = 1,
  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.
min_anchor	Integer specifiying minimum anchor length
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

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)
```

---

predictTxFeaturesPerSample

Identification of splice junctions and exons from BAM file

Description

Splice junctions and exons are predicted from genomic RNA-seq read alignments in BAM format.

Usage

```
predictTxFeaturesPerSample(file_bam, which, paired_end, read_length,
  frag_length, lib_size, min_junction_count, alpha, psi, beta, gamma,
  min_anchor, include_counts, retain_coverage, junctions_only, max_complexity,
  sample_name, verbose, cores)
```

Arguments

file_bam	BAM file with genomic RNA-seq read alignments
which	GRanges of genomic regions to be considered for feature prediction, passed to ScanBamParam
paired_end	Logical, TRUE for paired-end data, FALSE for single-end data
read_length	Read length required for use with alpha
frag_length	Fragment length for paired-end data required for use with alpha

`lib_size` Number of aligned fragments required for use with `alpha`

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

`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_anchor` Integer specifiying minimum anchor length

`include_counts` Logical indicating whether counts of compatible fragments should be included in metadata column "N"

`retain_coverage` Logical indicating whether coverage for each exon should be retained as an `RleList` in metadata column "coverage". This allows filtering of features using more stringent criteria after the initial prediction.

`junctions_only` Logical indicating whether predictions should be limited to identification of splice junctions only

`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`.

`sample_name` Sample name used in messages

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

`cores` Number of cores available for parallel processing

Details

For spliced alignments, the direction of transcription is inferred from the XS tag in the BAM file and used to assign strand information to the read, or fragment for paired-end data.

Feature prediction is performed in two steps. First, splice junctions are identified from spliced alignments. Second, exons are identified based on regions that are flanked by splice junctions and show sufficient coverage with compatible reads.

Splice junctions implied by read alignments are filtered based on fragment count and splice frequency. The splice frequency at the splice donor (acceptor) is defined as x_J/x_D (x_J/x_A), where x_J is the number of fragments containing the splice junction, and x_D (x_A) is the number of fragments overlapping the exon/intron (intron/exon) boundary. Fragments overlapping the spliced boundary can be either spliced or extend into the intron. To be included in predicted features, splice junctions must have fragment count at least `min_junction_count` or FPKM at least `alpha`, and splice frequency at both donor and acceptor at least `psi`.

Regions between any pair of identified splice junctions with sufficient compatible read coverage are considered candidate internal exons. Read coverage for a candidate exon is computed based

on compatible fragments, i.e. fragments with matching (or missing) strand information and introns consistent with the exon under consideration. Candidate exons are included in predicted features if the minimum coverage is at least `beta` * number of junction-containing fragments for either flanking junctions.

Terminal exons are regions downstream or upstream of splice junctions with compatible fragment coverage at least `gamma` * number of junction-containing fragments.

Value

`TxFeatures` object

Author(s)

Leonard Goldstein

`predictTxFeaturesPerStrand`

Identification of splice junctions and exons for a given chromosome and strand

---

Description

Identification of splice junctions and exons for a given chromosome and strand.

Usage

```
predictTxFeaturesPerStrand(file_bam, paired_end, which, min_junction_count, psi,
  beta, gamma, min_anchor, include_counts, retain_coverage, junctions_only,
  max_complexity, sample_name, verbose)
```

Arguments

`file_bam`	BAM file with genomic RNA-seq read alignments
`paired_end`	Logical, `TRUE` for paired-end data, `FALSE` for single-end data
`which`	`GRanges` of genomic regions to be considered for feature prediction, passed to `ScanBamParam`
`min_junction_count`	Minimum fragment count required for a splice junction to be included. If specified, argument `alpha` is ignored.
`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_anchor`	Integer specifiying minimum anchor length
`include_counts`	Logical indicating whether counts of compatible fragments should be included in metadata column "N"

`retain_coverage`
: Logical indicating whether coverage for each exon should be retained as an `RleList` in metadata column "coverage". This allows filtering of features using more stringent criteria after the initial prediction.

`junctions_only`
: Logical indicating whether predictions should be limited to identification of splice junctions only

`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`.

`sample_name`
: Sample name used in messages

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

Value

`GRanges` of predicted features

Author(s)

Leonard Goldstein

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

Description

The effect of a splice variant is predicted for individual protein-coding transcripts.

Usage

```
predictVariantEffects(sgv, tx, genome, fix_start_codon = TRUE,
  output = c("short", "full"), cores = 1)
```

Arguments

`sgv`
: `SGVariants` object

`tx`
: `TxDb` object, or `GRangesList` of exons grouped by transcript with metadata columns `txName`, `geneName`, `cdsStart` and `cdsEnd` (by convention, cdsStart < cdsEnd for both strands). For import from GFF format, use function `importTranscripts`.

`genome`
: `BSgenome` object

`fix_start_codon`
: Logical indicating whether the annotated start codon should be considered fixed and the variant transcript should not be scanned for alternative start codons

`output`
: Character string indicating whether short results or full results (with additional columns) should be returned

`cores`
: Number of cores available for parallel processing

Value

`data.frame` with rows corresponding to a variant-transcript pair. The output includes columns for variant identifier, transcript name, gene name, type of alteration at the RNA and protein level, and variant description at the RNA and protein level in HGVS notation. For `output = "full"` additional columns are returned. These include the full-length RNA and protein sequence for the reference and variant transcript. Event start and end coordinates in the full output are 0- and 1-based, respectively (to allow for description of deletions). Coordinates for the last junction in a transcript refer to the last base of the second-to-last exon.

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)
```

`removeExonsIsolated` *Remove exons with no flanking splice junctions*

Description

Remove exons with no flanking splice junctions.

Usage

```
removeExonsIsolated(features)
```

Arguments

`features`	`TxFeatures` object

Value

`TxFeatures` object with filtered features

Author(s)

Leonard Goldstein

sgfc_ann — *Example splice graph feature counts (annotation-based)*

Description

Compatible counts and FPKMs for FBXO31 splice graph features, based on UCSC knownGene annotation.

Format

`SGFeatureCounts` object

Author(s)

Leonard Goldstein

sgfc_pred — *Example splice graph feature counts (predicted)*

Description

Compatible counts and FPKMs for FBXO31 splice graph features, predicted from example BAM files.

Format

`SGFeatureCounts` object

Author(s)

Leonard Goldstein

SGFeatureCounts — *Splice graph feature counts*

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" and "FPKM"

Value

SGFeatureCounts object

Author(s)

Leonard Goldstein

Examples

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

SGFeatures *Splice graph features*

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 value J, E, D, or 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

SGFeatures object

Author(s)

Leonard Goldstein

Examples

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

---

sgf_ann *Example splice graph features (annotation-based)*

Description

Splice graph features for FBXO31, based on UCSC knownGene annotation.

Format

SGFeatures object

Author(s)

Leonard Goldstein

sgf_pred — *Example splice graph features (predicted)*

Description

Splice graph features for FBXO31, predicted from example BAM files.

Format

`SGFeatures` object

Author(s)

Leonard Goldstein

SGSegments — *Splice graph segments*

Description

Creates an instance of S4 class `SGSegments` for storing splice graph segments.

Usage

```
SGSegments(x)
```

Arguments

x	`GRangesList` of `SGFeatures` with appropriate outer metadata columns

Value

`SGSegments` object

Author(s)

Leonard Goldstein

SGVariantCounts *Splice graph variant counts*

Description

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

Usage

```
SGVariantCounts(x)
```

Arguments

x	`RangedSummarizedExperiment` with `SGVariants` as `rowRanges` and assays "variantFreq", "countsVariant5p", "countsVariant3p", "countsEvent5p", "countsEvent3p", and optionally "countsVariant5pOr3p"

Value

`SGVariantCounts` object

Author(s)

Leonard Goldstein

Examples

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

SGVariants *Splice graph variants*

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

Details

SGVariants includes columns as described below.

- from and to indicate the variant start and end, respectively. from nodes are splice donors ("D") or transcript starts ("S"). to nodes are splice acceptors ("A") or transcript ends ("E").
- type and featureID describe the variant in terms of the splice graph features that make up the variant.
- segmentID specifies unique identifiers labelling unbranched segments of the splice graph.
- closed5p indicates whether nodes in the variant can be reached from nodes outside of the variant exclusively through the from node.
- closed3p indicates whether nodes in the variant can reach nodes outside of the variant exclusively through the to node.
- closed5pEvent indicates whether nodes in the event can be reached from nodes outside of the event exclusively through the from node.
- closed3pEvent indicates whether nodes in the event can reach nodes outside of the event exclusively through the to node.
- geneID has the same interpretation as for SGFeatures.
- eventID and variantID are unique identifiers for each event and variant, respectively.
- featureID5p and featureID3p indicate representative features used for variant quantification at the start and end of the variant, respectively.
- featureID5pEvent and featureID3pEvent indicate the ensemble of representative features at the start and end of the event, respectively.
- txName indicates structurally compatible transcripts.
- geneName behaves as for SGFeatures.
- variantType indicates whether a splice variant is consistent with a canonical splice event (for a list of possible values, see the manual page for annotateSGVariants).
- variantName provides a unique name for each splice variant (for details, see the manual page for makeVariantNames).

Value

SGVariants object

Author(s)

Leonard Goldstein

Examples

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

sgvc_ann — *Example splice variant counts (annotated)*

Description

Splice variant counts and frequencies for FBXO31. Splice variants are based on UCSC knownGene annotation.

Format

`SGVariantCounts` object

Author(s)

Leonard Goldstein

sgvc_ann_from_bam — *Example splice variant counts (annotated) from BAM files*

Description

Splice variant counts and frequencies for FBXO31. Splice variants are based on UCSC knownGene annotation. Counts were obtained from BAM files.

Format

`SGVariantCounts` object

Author(s)

Leonard Goldstein

sgvc_pred — *Example splice variant counts (predicted)*

Description

Splice variant counts and frequencies for FBXO31. Splice variants were predicted from example BAM files.

Format

`SGVariantCounts` object

Author(s)

Leonard Goldstein

sgvc_pred_from_bam *Example splice variant counts (predicted) from BAM files*

Description

Splice variant counts and frequencies for FBXO31. Splice variants were predicted from example BAM files. Counts were obtained from BAM files.

Format

`SGVariantCounts` object

Author(s)

Leonard Goldstein

sgv_ann *Example splice variants (annotation-based)*

Description

Splice variants for FBXO31, based on UCSC knownGene annotation.

Format

`SGVariants` object

Author(s)

Leonard Goldstein

sgv_pred *Example splice variants (predicted)*

Description

Splice variants for FBXO31, predicted from example BAM files.

Format

`SGVariants` object

Author(s)

Leonard Goldstein

si — *Example sample information*

Description

Sample information for example BAM files included in the `SGSeq` package.

Format

`data.frame` with columns "sample_name", "file_bam", "paired_end", "read_length", "frag_length" and "lib_size".

Author(s)

Leonard Goldstein

slots — *Accessing and replacing metadata columns*

Description

Accessor and replacement functions for metadata columns.

Usage

```
type(x) <- value

txName(x)

txName(x) <- value

geneName(x)

geneName(x) <- value

featureID(x)

featureID(x) <- value

geneID(x)

geneID(x) <- value

splice5p(x)
```

```
splice5p(x) <- value

splice3p(x)

splice3p(x) <- value

from(x) <- value

to(x) <- value

segmentID(x)

segmentID(x) <- value

variantID(x)

variantID(x) <- value

eventID(x)

eventID(x) <- value

closed5p(x)

closed5p(x) <- value

closed3p(x)

closed3p(x) <- value

closed5pEvent(x)

closed5pEvent(x) <- value

closed3pEvent(x)

closed3pEvent(x) <- value

variantType(x)

variantType(x) <- value

variantName(x)

variantName(x) <- value

featureID5p(x)
```

```
featureID5p(x) <- value

featureID3p(x)

featureID3p(x) <- value

featureID5pEvent(x)

featureID5pEvent(x) <- value

featureID3pEvent(x)

featureID3pEvent(x) <- value

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

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

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

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

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

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

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

## S4 method for signature 'Paths'
txName(x)

## S4 method for signature 'Counts'
txName(x)

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

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

## S4 replacement method for signature 'Counts'
```

```
txName(x) <- value

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

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

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

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

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

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

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

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

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

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

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

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

## S4 method for signature 'SGFeatures'
geneID(x)

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

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

## S4 replacement method for signature 'SGFeatures'
```

```
geneID(x) <- value

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## S4 method for signature 'SGVariantCounts'
```

```
segmentID(x)

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

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

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

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

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

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

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

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

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

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

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

## S4 method for signature 'SGVariantCounts'
eventID(x)

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

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

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

## S4 method for signature 'SGVariantCounts'
```

```
variantID(x)

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

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

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

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

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

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

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

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

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

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

## S4 method for signature 'SGVariants'
closed5pEvent(x)

## S4 method for signature 'SGVariantCounts'
closed5pEvent(x)

## S4 replacement method for signature 'SGVariants'
closed5pEvent(x) <- value

## S4 replacement method for signature 'SGVariantCounts'
closed5pEvent(x) <- value

## S4 method for signature 'SGVariants'
closed3pEvent(x)

## S4 method for signature 'SGVariantCounts'
```

```
closed3pEvent(x)

## S4 replacement method for signature 'SGVariants'
closed3pEvent(x) <- value

## S4 replacement method for signature 'SGVariantCounts'
closed3pEvent(x) <- value

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

## S4 method for signature 'SGVariantCounts'
variantName(x)

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

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

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

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

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

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

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

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

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

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

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

## S4 method for signature 'SGVariantCounts'
```

```
featureID3p(x)

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

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

## S4 method for signature 'SGVariants'
featureID5pEvent(x)

## S4 method for signature 'SGVariantCounts'
featureID5pEvent(x)

## S4 replacement method for signature 'SGVariants'
featureID5pEvent(x) <- value

## S4 replacement method for signature 'SGVariantCounts'
featureID5pEvent(x) <- value

## S4 method for signature 'SGVariants'
featureID3pEvent(x)

## S4 method for signature 'SGVariantCounts'
featureID3pEvent(x)

## S4 replacement method for signature 'SGVariants'
featureID3pEvent(x) <- value

## S4 replacement method for signature 'SGVariantCounts'
featureID3pEvent(x) <- value
```

Arguments

`x`	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.

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))
```

---

splicesiteOverlap *Compatible fragment counts for splice sites*

Description

Identify fragments with alignments extending across exon/intron boundaries.

Usage

```
splicesiteOverlap(splicesites, side, frag_exonic, frag_intron, min_anchor,
  include = c("all", "spliced", "unspliced"), counts = TRUE)
```

Arguments

`splicesites`	`IRanges` of splice sites
`side`	Character vector indicating whether the spliced boundary is to the left ("L") or right ("R") of the splice site
`frag_exonic`	`IRangesList` of exonic regions, one entry per fragment
`frag_intron`	`IRangesList` of introns, one entry per fragment
`min_anchor`	Integer specifiying minimum anchor length
`include`	Character string indicating whether considered fragments should be all that overlap the splice site ("all"), those that are spliced at the site ("spliced") or those that are not spliced, i.e. extend into the adjacent intron ("unspliced")
`counts`	Logical indicating whether counts or indices of compatible fragments should be returned

Value

Counts or list of indices of compatible fragments

Author(s)

Leonard Goldstein

tx — *Example transcripts*

Description

FBXO31 transcripts, based on UCSC knownGene annotation. Suitable as input for `convertToTxFeatures` and `predictVariantEffects`.

Format

`GRangesList` object

Author(s)

Leonard Goldstein

TxFeatures — *Transcript features*

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 value `J`, `I`, `F`, `L`, or `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

`TxFeatures` object

Author(s)

Leonard Goldstein

Examples

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

---

txf_ann *Example transcript features (annotation-based)*

Description

Transcript features for FBXO31, based on UCSC knownGene annotation.

Format

`TxFeatures` object

Author(s)

Leonard Goldstein

---

txf_pred *Example transcript features (predicted)*

Description

Transcript features for FBXO31, predicted from example BAM files.

Format

`TxFeatures` object

Author(s)

Leonard Goldstein

updateObject *Update object*

Description

Update object created with previous version of SGSeq.

Usage

```
## S4 method for signature 'SGVariants'
updateObject(object, ..., verbose = FALSE)

## S4 method for signature 'SGVariantCounts'
updateObject(object, ..., verbose = FALSE)
```

Arguments

object	Object to be updated
...	Additional arguments
verbose	Should a warning message be generated

Value

Updated object

Author(s)

Leonard Goldstein

Index

∗ **internal**
- annotateSGVariants, 7
- exonCompatible, 11
- filterFeatures, 13
- findOverlapsRanges, 14
- getSGFeatureCountsPerSample, 17
- gr, 19
- junctionCompatible, 20
- makeVariantNames, 21
- predictCandidatesInternal, 29
- predictCandidatesTerminal, 30
- predictExonsInternal, 30
- predictExonsTerminal, 31
- predictJunctions, 32
- predictSpliced, 33
- predictTxFeaturesPerSample, 35
- predictTxFeaturesPerStrand, 37
- removeExonsIsolated, 40
- sgf_ann, 43
- sgf_pred, 44
- sgfc_ann, 41
- sgfc_pred, 41
- SGSegments, 44
- sgv_ann, 48
- sgv_pred, 48
- sgvc_ann, 47
- sgvc_ann_from_bam, 47
- sgvc_pred, 47
- sgvc_pred_from_bam, 48
- si, 49
- splicesiteOverlap, 58
- tx, 59
- txf_ann, 60
- txf_pred, 60

analyzeFeatures, 3
analyzeVariants, 5
annotate, *4*, 6
annotateSGVariants, 7, *15*
assays, 8
closed3p *(slots)*, 49
closed3p,SGVariantCounts-method *(slots)*, 49
closed3p,SGVariants-method *(slots)*, 49
closed3p<- *(slots)*, 49
closed3p<-,SGVariantCounts-method *(slots)*, 49
closed3p<-,SGVariants-method *(slots)*, 49
closed3pEvent *(slots)*, 49
closed3pEvent,SGVariantCounts-method *(slots)*, 49
closed3pEvent,SGVariants-method *(slots)*, 49
closed3pEvent<- *(slots)*, 49
closed3pEvent<-,SGVariantCounts-method *(slots)*, 49
closed3pEvent<-,SGVariants-method *(slots)*, 49
closed5p *(slots)*, 49
closed5p,SGVariantCounts-method *(slots)*, 49
closed5p,SGVariants-method *(slots)*, 49
closed5p<- *(slots)*, 49
closed5p<-,SGVariantCounts-method *(slots)*, 49
closed5p<-,SGVariants-method *(slots)*, 49
closed5pEvent *(slots)*, 49
closed5pEvent,SGVariantCounts-method *(slots)*, 49
closed5pEvent,SGVariants-method *(slots)*, 49
closed5pEvent<- *(slots)*, 49
closed5pEvent<-,SGVariantCounts-method *(slots)*, 49
closed5pEvent<-,SGVariants-method *(slots)*, 49
convertToSGFeatures, *4*, 9
convertToTxFeatures, 10
counts,SGFeatureCounts-method *(assays)*,

8
counts,SGVariantCounts-method (assays), 8
counts<-,SGFeatureCounts-method (assays), 8
counts<-,SGVariantCounts-method (assays), 8

eventID (slots), 49
eventID,SGVariantCounts-method (slots), 49
eventID,SGVariants-method (slots), 49
eventID<- (slots), 49
eventID<-,SGVariantCounts-method (slots), 49
eventID<-,SGVariants-method (slots), 49
exonCompatible, 11
exportFeatures, 12

featureID (slots), 49
featureID,Counts-method (slots), 49
featureID,Paths-method (slots), 49
featureID,SGFeatures-method (slots), 49
featureID3p (slots), 49
featureID3p,SGVariantCounts-method (slots), 49
featureID3p,SGVariants-method (slots), 49
featureID3p<- (slots), 49
featureID3p<-,SGVariantCounts-method (slots), 49
featureID3p<-,SGVariants-method (slots), 49
featureID3pEvent (slots), 49
featureID3pEvent,SGVariantCounts-method (slots), 49
featureID3pEvent,SGVariants-method (slots), 49
featureID3pEvent<- (slots), 49
featureID3pEvent<-,SGVariantCounts-method (slots), 49
featureID3pEvent<-,SGVariants-method (slots), 49
featureID5p (slots), 49
featureID5p,SGVariantCounts-method (slots), 49
featureID5p,SGVariants-method (slots), 49
featureID5p<- (slots), 49
featureID5p<-,SGVariantCounts-method (slots), 49
featureID5p<-,SGVariants-method (slots), 49
featureID5pEvent (slots), 49
featureID5pEvent,SGVariantCounts-method (slots), 49
featureID5pEvent,SGVariants-method (slots), 49
featureID5pEvent<- (slots), 49
featureID5pEvent<-,SGVariantCounts-method (slots), 49
featureID5pEvent<-,SGVariants-method (slots), 49
featureID<- (slots), 49
featureID<-,Counts-method (slots), 49
featureID<-,Paths-method (slots), 49
featureID<-,SGFeatures-method (slots), 49
filterFeatures, 13
findOverlapsRanges, 14
findSGVariants, 5, 14
FPKM (assays), 8
FPKM,SGFeatureCounts-method (assays), 8
FPKM,SGVariantCounts-method (assays), 8
FPKM<- (assays), 8
FPKM<-,SGFeatureCounts-method (assays), 8
from,Paths-method (slots), 49
from,SGVariantCounts-method (slots), 49
from<- (slots), 49
from<-,Paths-method (slots), 49
from<-,SGVariantCounts-method (slots), 49

geneID (slots), 49
geneID,Counts-method (slots), 49
geneID,Paths-method (slots), 49
geneID,SGFeatures-method (slots), 49
geneID<- (slots), 49
geneID<-,Counts-method (slots), 49
geneID<-,Paths-method (slots), 49
geneID<-,SGFeatures-method (slots), 49
geneName (slots), 49
geneName,Counts-method (slots), 49
geneName,Features-method (slots), 49
geneName,Paths-method (slots), 49
geneName<- (slots), 49
geneName<-,Counts-method (slots), 49

geneName<-,Features-method *(slots)*, 49
geneName<-,Paths-method *(slots)*, 49
getBamInfo, 15
getSGFeatureCounts, *4*, 16
getSGFeatureCountsPerSample, 17
getSGVariantCounts, *5*, 18
gr, 19

importTranscripts, 19

junctionCompatible, 20

makeSGFeatureCounts, 21
makeVariantNames, 21
mergeTxFeatures, 22, *34*

plotCoverage, 23
plotFeatures, 24
plotSpliceGraph, 26
plotVariants, 28
predictCandidatesInternal, 29
predictCandidatesTerminal, 30
predictExonsInternal, 30
predictExonsTerminal, 31
predictJunctions, 32
predictSpliced, 33
predictTxFeatures, *4*, 34
predictTxFeaturesPerSample, *34*, 35
predictTxFeaturesPerStrand, 37
predictVariantEffects, 38
processTerminalExons, *34*, 39

removeExonsIsolated, 40

segmentID *(slots)*, 49
segmentID,Paths-method *(slots)*, 49
segmentID,SGVariantCounts-method *(slots)*, 49
segmentID<- *(slots)*, 49
segmentID<-,Paths-method *(slots)*, 49
segmentID<-,SGVariantCounts-method *(slots)*, 49
sgf_ann, 43
sgf_pred, 44
sgfc_ann, 41
sgfc_pred, 41
SGFeatureCounts, 41
SGFeatures, 42
SGSegments, 44
sgv_ann, 48
sgv_pred, 48
SGVariantCounts, 45
SGVariants, 45
sgvc_ann, 47
sgvc_ann_from_bam, 47
sgvc_pred, 47
sgvc_pred_from_bam, 48
si, 49
slots, 49
splice3p *(slots)*, 49
splice3p,SGFeatureCounts-method *(slots)*, 49
splice3p,SGFeatures-method *(slots)*, 49
splice3p,SGSegments-method *(slots)*, 49
splice3p<- *(slots)*, 49
splice3p<-,SGFeatureCounts-method *(slots)*, 49
splice3p<-,SGFeatures-method *(slots)*, 49
splice3p<-,SGSegments-method *(slots)*, 49
splice5p *(slots)*, 49
splice5p,SGFeatureCounts-method *(slots)*, 49
splice5p,SGFeatures-method *(slots)*, 49
splice5p,SGSegments-method *(slots)*, 49
splice5p<- *(slots)*, 49
splice5p<-,SGFeatureCounts-method *(slots)*, 49
splice5p<-,SGFeatures-method *(slots)*, 49
splice5p<-,SGSegments-method *(slots)*, 49
splicesiteOverlap, 58

to,Paths-method *(slots)*, 49
to,SGVariantCounts-method *(slots)*, 49
to<- *(slots)*, 49
to<-,Paths-method *(slots)*, 49
to<-,SGVariantCounts-method *(slots)*, 49
tx, 59
txf_ann, 60
txf_pred, 60
TxFeatures, 59
txName *(slots)*, 49
txName,Counts-method *(slots)*, 49
txName,Features-method *(slots)*, 49
txName,Paths-method *(slots)*, 49
txName<- *(slots)*, 49
txName<-,Counts-method *(slots)*, 49
txName<-,Features-method *(slots)*, 49
txName<-,Paths-method *(slots)*, 49
type,Counts-method *(slots)*, 49

type,Features-method (slots), 49
type,Paths-method (slots), 49
type<- (slots), 49
type<-,Counts-method (slots), 49
type<-,Features-method (slots), 49
type<-,Paths-method (slots), 49

updateObject, 61
updateObject,SGVariantCounts-method (updateObject), 61
updateObject,SGVariants-method (updateObject), 61

variantFreq (assays), 8
variantFreq,SGVariantCounts-method (assays), 8
variantFreq<- (assays), 8
variantFreq<-,SGVariantCounts-method (assays), 8
variantID (slots), 49
variantID,SGVariantCounts-method (slots), 49
variantID,SGVariants-method (slots), 49
variantID<- (slots), 49
variantID<-,SGVariantCounts-method (slots), 49
variantID<-,SGVariants-method (slots), 49
variantName (slots), 49
variantName,SGVariantCounts-method (slots), 49
variantName,SGVariants-method (slots), 49
variantName<- (slots), 49
variantName<-,SGVariantCounts-method (slots), 49
variantName<-,SGVariants-method (slots), 49
variantType (slots), 49
variantType,SGVariantCounts-method (slots), 49
variantType,SGVariants-method (slots), 49
variantType<- (slots), 49
variantType<-,SGVariantCounts-method (slots), 49
variantType<-,SGVariants-method (slots), 49