Package 'GeneStructureTools'

April 15, 2020

Type Package

Title Tools for spliced gene structure manipulation and analysis

Version 1.6.0

Author Beth Signal

Maintainer Beth Signal <b.signal@garvan.org.au>

Description GeneStructureTools can be used to create in silico alternative splicing events, and analyse potential effects this has on functional gene products.

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 6.0.1

Imports Biostrings,GenomicRanges,IRanges,data.table,plyr,stringdist,stringr,S4Vectors,BSgenome.Mmusculus.UCSC.mm10,s

biocViews ImmunoOncology, Software, DifferentialSplicing, FunctionalPrediction, Transcriptomics, AlternativeSplicing, RNASeq

Suggests BiocStyle, knitr, rmarkdown

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

git_branch RELEASE_3_10

git_last_commit 94b91cf

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

R topics documented:

`addBroadTypes` *Change transcript biotypes to a broader set*

Description

Change transcript biotypes to a broader set in a GRanges GTF object

Usage

```
addBroadTypes(gtf)
```

Arguments

`gtf`	GRanges object of the GTF

Value

GRanges object of the GTF with new transcript types

Author(s)

Beth Signal

See Also

Other gtf manipulation: `UTR2UTR53`, `exonsToTranscripts`, `filterGtfOverlap`, `removeDuplicateTranscripts`, `removeSameExon`, `reorderExonNumbers`

Examples

```
gtfFile <- system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools")
gtf <- rtracklayer::import(gtfFile)
gtf <- addBroadTypes(gtf)
```

---

addIntronInTranscript *Add a retained intron to the transcripts it is skipped by*

Description

Add a retained intron to the transcripts it is skipped by

Usage

```
addIntronInTranscript(flankingExons, exons, whippetDataSet = NULL,
  match = "exact", glueExons = TRUE)
```

Arguments

`flankingExons`	data.frame generataed by findIntronContainingTranscripts()
`exons`	GRanges object made from a GTF with ONLY exon annotations (no gene, transcript, CDS etc.)
`whippetDataSet`	whippetDataSet generated from `readWhippetDataSet()`
`match`	what type of match replacement should be done? exact: exact matches to the intron only retain: keep non-exact intron match coordinates in spliced sets, and retain them in retained sets replace: replace non-exact intron match coordinates with event coordinates in spliced sets, and retain in retained sets
`glueExons`	Join together exons that are not seperated by introns?

Value

GRanges with transcripts containing retained introns

Author(s)

Beth Signal

See Also

Other whippet splicing isoform creation: findExonContainingTranscripts, findIntronContainingTranscripts, findJunctionPairs, replaceJunction, skipExonInTranscript

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.intronRetention <- filterWhippetEvents(wds, eventTypes="RI")
exons.intronRetention <- findIntronContainingTranscripts(wds.intronRetention, exons)
IntronRetentionTranscripts <- addIntronInTranscript(exons.intronRetention, exons,
whippetDataSet=wds.intronRetention)

exonsFromGRanges <- exons[exons$transcript_id=="ENSMUST00000139129.8" &
exons$exon_number %in% c(3,4)]
intronFromGRanges <- exonsFromGRanges[1]
GenomicRanges::start(intronFromGRanges) <-
GenomicRanges::end(exonsFromGRanges[exonsFromGRanges$exon_number==3])
GenomicRanges::end(intronFromGRanges) <-
GenomicRanges::start(exonsFromGRanges[exonsFromGRanges$exon_number==4])
exons.intronRetention <- findIntronContainingTranscripts(intronFromGRanges, exons)

IntronRetentionTranscripts <-
addIntronInTranscript(exons.intronRetention, exons, match="retain")
```

---

alternativeIntronUsage

Create transcripts with alternative intron usage

Description

Creates transcript isoforms from alternative intron usage tested by leafcutter

Usage

```
alternativeIntronUsage(altIntronLocs, exons)
```

Arguments

altIntronLocs	data.frame containing information from the per_intron_results.tab file output from leafcutter. Note that only one cluster of alternative introns can be processed at a time.
exons	GRanges object made from a GTF with ONLY exon annotations (no gene, transcript, CDS etc.)

Value

GRanges object with all potential alternative isoforms skipping the introns specified in either the upregulated or downregulated locations

Author(s)

Beth Signal

Examples

```
leafcutterFiles <- list.files(system.file("extdata","leafcutter/",
package = "GeneStructureTools"), full.names = TRUE)
leafcutterIntrons <- read.delim(leafcutterFiles[grep("intron_results",
leafcutterFiles)],stringsAsFactors=FALSE)
gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
# single cluster processing
cluster <- leafcutterIntrons[leafcutterIntrons$cluster=="chr16:clu_1396",]
altIsoforms1396 <- alternativeIntronUsage(cluster, exons)
unique(altIsoforms1396$transcript_id)
cluster <- leafcutterIntrons[leafcutterIntrons$cluster=="chr16:clu_1395",]
altIsoforms1395 <- alternativeIntronUsage(cluster, exons)
unique(altIsoforms1395$transcript_id)
# multiple cluster processing
altIsoforms1396plus1395 <- alternativeIntronUsage(cluster, c(exons, altIsoforms1396))
unique(altIsoforms1396plus1395$transcript_id)
```

---

annotateGeneModel *Annotate a GRanges gene model with ORF boundries for visualisation with Gviz*

Description

Annotate a GRanges gene model with ORF boundries for visualisation with Gviz

Usage

```
annotateGeneModel(transcripts, orfs)
```

Arguments

transcripts	GRanges of gene model to be visualised
orfs	ORF predictions. Created by getORFs()

Value

data.frame of a gene model for visualisation

Author(s)

Beth Signal

See Also

Other Gviz gene structure visualisation: makeGeneModel

Examples

```
gtf <- rtracklayer::import(system.file("extdata", "example_gtf.gtf",
package="GeneStructureTools"))
transcript <- gtf[gtf$type=="exon" & gtf$gene_name=="Neurl1a"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
# longest ORF for each transcripts
orfs <- getOrfs(transcript, BSgenome = g, returnLongestOnly = TRUE)
geneModelAnnotated <- annotateGeneModel(transcript, orfs)
```

---

attrChangeAltSpliced *Evaluate the change in an attribute between a set of 'normal' transcripts and 'alternative' transcripts*

Description

Evaluate the change in an attribute between a set of 'normal' transcripts and 'alternative' transcripts

Usage

```
attrChangeAltSpliced(orfsX, orfsY, attribute = "orf_length",
  compareBy = "gene", useMax = TRUE, compareUTR = FALSE)
```

Arguments

orfsX	orf information for 'normal' transcripts. Generated by getOrfs()
orfsY	orf information for 'alternative' transcripts. Generated by getOrfs()
attribute	attribute to compare
compareBy	compare by 'transcript' isoforms or by 'gene' groups
useMax	use max as the summary function when multiple isoforms are aggregated? If FALSE, will use min instead.
compareUTR	compare the UTR lengths between transcripts? Only runs if attribute = "orf_length"

Value

data.frame with attribute changes

Author(s)

Beth Signal

See Also

Other transcript isoform comparisons: orfDiff, transcriptChangeSummary

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
transcripts <- gtf[gtf$type=="transcript"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.exonSkip <- filterWhippetEvents(wds, eventTypes="CE",psiDelta = 0.2)
exons.exonSkip <- findExonContainingTranscripts(wds.exonSkip, exons,
variableWidth=0, findIntrons=FALSE, transcripts)
ExonSkippingTranscripts <- skipExonInTranscript(exons.exonSkip, exons, whippetDataSet=wds.exonSkip)

orfsSkipped <- getOrfs(ExonSkippingTranscripts[ExonSkippingTranscripts$set=="skipped_exon"],
BSgenome = g)
orfsIncluded <- getOrfs(ExonSkippingTranscripts[ExonSkippingTranscripts$set=="included_exon"],
BSgenome = g)
attrChangeAltSpliced(orfsSkipped, orfsIncluded,attribute = "orf_length")
```

`coordinates` *Method coordinates*

Description

Method coordinates

Usage

```
coordinates(whippetDataSet)

## S4 method for signature 'whippetDataSet'
coordinates(whippetDataSet)
```

Arguments

`whippetDataSet`	whippetDataSet generated from `readWhippetDataSet()`

Value

whippet splicing event coordinates as a GRanges object

See Also

Other whippet data processing: `diffSplicingResults`, `filterWhippetEvents`, `formatWhippetEvents`, `junctions`, `readCounts`, `readWhippetDIFFfiles`, `readWhippetDataSet`, `readWhippetJNCfiles`, `readWhippetPSIfiles`, `whippetTranscriptChangeSummary`

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)

coordinates <- coordinates(wds)
```

`DEXSeqIdsToGeneIds` *Convert DEXSeq ids to gene ids*

Description

Convert DEXSeq ids to gene ids

Usage

```
DEXSeqIdsToGeneIds(DEXSeqIds, removeVersion = FALSE, containsE = TRUE)
```

Arguments

`DEXSeqIds`	vector of DEXSeq group or exon ids
`removeVersion`	remove the version (.xx) of the gene?
`containsE`	do the DEXSeq exons ids contain :E00X?

Value

vector of unique gene ids

Author(s)

Beth Signal

See Also

Other DEXSeq processing methods: `findDEXexonType`, `summariseExonTypes`

Examples

```
# multiple genes in name
DEXSeqId <- "ENSMUSG00000027618.17+ENSMUSG00000098950.7+ENSMUSG00000089824.10+ENSMUSG00000074643.12"
DEXSeqIdsToGeneIds(DEXSeqId)

# exonic part number in id
DEXSeqIdsToGeneIds("ENSMUSG00000001017.15:E013", removeVersion=TRUE)
```

diffSplicingResults *Method diffSplicingResults*

Description

Method diffSplicingResults

Usage

```
diffSplicingResults(whippetDataSet)

## S4 method for signature 'whippetDataSet'
diffSplicingResults(whippetDataSet)
```

Arguments

whippetDataSet	whippetDataSet generated from readWhippetDataSet()

Value

differential splicing results data.frame (originally from a whippet .diff file)

See Also

Other whippet data processing: coordinates, filterWhippetEvents, formatWhippetEvents, junctions, readCounts, readWhippetDIFFfiles, readWhippetDataSet, readWhippetJNCfiles, readWhippetPSIfiles, whippetTranscriptChangeSummary

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)

diffSplicingResults <- diffSplicingResults(wds)
```

exonsToTranscripts *Convert an exon-level gtf annotation to a transcript-level gtf annotation*

Description

Convert an exon-level gtf annotation to a transcript-level gtf annotation

Usage

```
exonsToTranscripts(exons)
```

Arguments

exons	GRanges object with exons

Value

GRanges object with transcripts

Author(s)

Beth Signal

See Also

Other gtf manipulation: UTR2UTR53, addBroadTypes, filterGtfOverlap, removeDuplicateTranscripts, removeSameExon, reorderExonNumbers

Examples

```
gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon" & gtf$transcript_id=="ENSMUST00000126412.1"]
exons
transcripts <- exonsToTranscripts(exons)
transcripts
```

`filterGtfOverlap` *Filter a GTF overlap to remove exons when exon is annotated as a CDS/UTR*

Description

Filter a GTF overlap to remove exons when exon is annotated as a CDS/UTR

Usage

```
filterGtfOverlap(gtf.from)
```

Arguments

`gtf.from`	GRanges object of the GTF produced from an overlap

Value

GRanges object of the GTF with redundant exons removed

Author(s)

Beth Signal

See Also

Other gtf manipulation: UTR2UTR53, addBroadTypes, exonsToTranscripts, removeDuplicateTranscripts, removeSameExon, reorderExonNumbers

Examples

```
gtfFile <- system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools")
gtf <- rtracklayer::import(gtfFile)
overlap <- as.data.frame(GenomicRanges::findOverlaps(gtf[which(gtf$type=="CDS")[1]], gtf))
table(gtf$type[overlap$subjectHits])
overlapFiltered <- filterGtfOverlap(gtf[overlap$subjectHits])
table(overlapFiltered$type[overlap$subjectHits])
overlap <- as.data.frame(GenomicRanges::findOverlaps(gtf[which(
gtf$transcript_type=="retained_intron")[1]],gtf))
table(gtf$type[overlap$subjectHits])
overlapFiltered <- filterGtfOverlap(gtf[overlap$subjectHits])
table(overlapFiltered$type[overlap$subjectHits])
```

`filterWhippetEvents` *Filter out significant events from a whippet diff comparison*

Description

Filter out significant events from a whippet diff comparison

Usage

```
filterWhippetEvents(whippetDataSet, probability = 0.95, psiDelta = 0.1,
  eventTypes = "all", idList = NA, minCounts = NA, medianCounts = NA,
  sampleTable)
```

Arguments

`whippetDataSet`	whippetDataSet generated from `readWhippetDataSet()`
`probability`	minimum probability required to call event as significant
`psiDelta`	minimum change in psi required to call an event as significant
`eventTypes`	which event type to filter for? default = `"all"`
`idList`	(optional) list of gene ids to filter for
`minCounts`	minumum number of counts for all replicates in at least one condition to call an event as significant
`medianCounts`	median count for all replicates in at least one condition to call an event as significant
`sampleTable`	data.frame with sample names and conditions. Only needed if filtering with counts.

Value

filtered whippet differential comparison data.frame

Author(s)

Beth Signal

See Also

Other whippet data processing: coordinates, diffSplicingResults, formatWhippetEvents, junctions, readCounts, readWhippetDIFFfiles, readWhippetDataSet, readWhippetJNCfiles, readWhippetPSIfiles, whippetTranscriptChangeSummary

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)
```

`findDEXexonType` *Find a DEXSeq exons' biotype*

Description

Find a DEXSeq exons' biotype

Usage

```
findDEXexonType(DEXSeqExonId, DEXSeqGtf, gtf, set = "overlap")
```

Arguments

DEXSeqExonId	vector of DEXSeq exon ids
DEXSeqGtf	GRanges object of the DEXSeq formatted gtf
gtf	GRanges object of the GTF annotated with exon biotypes - i.e. exon, CDS, UTR
set	which overlapping set of exon biotypes to return - to, from, and/or overlap

Value

overlaping types

Author(s)

Beth Signal

See Also

Other DEXSeq processing methods: DEXSeqIdsToGeneIds, summariseExonTypes

Examples

```
gtfFile <- system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools")
DEXSeqGtfFile <- system.file("extdata","gencode.vM14.dexseq.gtf",
package = "GeneStructureTools")

gtf <- rtracklayer::import(gtfFile)
gtf <- UTR2UTR53(gtf)
DEXSeqGtf <- rtracklayer::import(DEXSeqGtfFile)
```

```
findDEXexonType("ENSMUSG00000032366.15:E028", DEXSeqGtf, gtf)

DEXSeqResultsFile <- system.file("extdata","dexseq_results_significant.txt",
package = "GeneStructureTools")
DEXSeqResults <- read.table(DEXSeqResultsFile, sep="\t")

findDEXexonType(rownames(DEXSeqResults), DEXSeqGtf, gtf)
```

`findExonContainingTranscripts` *Given the location of a whole spliced in exon, find transcripts which can splice out this exon*

Description

Given the location of a whole spliced in exon, find transcripts which can splice out this exon

Usage

```
findExonContainingTranscripts(input, exons, variableWidth = 0,
  findIntrons = FALSE, transcripts)
```

Arguments

`input`	whippetDataSet generated from `readWhippetDataSet()` or a Granges of exon coordinates
`exons`	GRanges object made from a GTF containing exon coordinates
`variableWidth`	How many nts overhang is allowed for finding matching exons (default = 0, i.e. complete match)
`findIntrons`	Find transcripts where the event occurs within the intron?
`transcripts`	GRanges object made from a GTF containing transcript coordinates (only required if findIntrons=TRUE)

Value

data.frame with all overlapping exons

Author(s)

Beth Signal

See Also

Other whippet splicing isoform creation: `addIntronInTranscript`, `findIntronContainingTranscripts`, `findJunctionPairs`, `replaceJunction`, `skipExonInTranscript`

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
transcripts <- gtf[gtf$type=="transcript"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.exonSkip <- filterWhippetEvents(wds, eventTypes="CE",psiDelta = 0.2)
exons.exonSkip <- findExonContainingTranscripts(wds.exonSkip, exons,
variableWidth=0, findIntrons=FALSE, transcripts)

exonFromGRanges <- exons[exons$exon_id == "ENSMUSE00001271768.1"]
exons.exonSkip <- findExonContainingTranscripts(exonFromGRanges, exons,
variableWidth=0, findIntrons=FALSE, transcripts)
```

---

findIntronContainingTranscripts

Given the location of a whole retained intron, find transcripts which splice out this intron

---

Description

Given the location of a whole retained intron, find transcripts which splice out this intron

Usage

```
findIntronContainingTranscripts(input, exons, match = "exact")
```

Arguments

input	whippetDataSet generated from `readWhippetDataSet()` or a Granges of intron coordinates
exons	GRanges object made from a GTF with ONLY exon annotations (no gene, transcript, CDS etc.)
match	what type of matching to perform? exact = only exons which bound the intron exactly, introns = any exon pairs which overlap the intron, all = any exon pairs AND single exons which overlap the intron

Value

data.frame with all flanking exon pairs

Author(s)

Beth Signal

See Also

Other whippet splicing isoform creation: addIntronInTranscript, findExonContainingTranscripts, findJunctionPairs, replaceJunction, skipExonInTranscript

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.intronRetention <- filterWhippetEvents(wds, eventTypes="RI")
exons.intronRetention <- findIntronContainingTranscripts(input=wds.intronRetention, exons)

exonsFromGRanges <- exons[exons$transcript_id=="ENSMUST00000139129.8" &
exons$exon_number %in% c(3,4)]
intronFromGRanges <- exonsFromGRanges[1]
GenomicRanges::start(intronFromGRanges) <-
GenomicRanges::end(exonsFromGRanges[exonsFromGRanges$exon_number==3])
GenomicRanges::end(intronFromGRanges) <-
GenomicRanges::start(exonsFromGRanges[exonsFromGRanges$exon_number==4])
exons.intronRetention <- findIntronContainingTranscripts(intronFromGRanges, exons)
```

`findJunctionPairs` *Find alternative junctions for Whippet alternative splicing events*

Description

Find junctions that pair with each end of an AA (alt. acceptor) or AD (alt. donor) whippet range Find junctions that pair with the upsteam/downstream exon of an AF (alt. first exon) or an AL (alt. last exon)

Usage

```
findJunctionPairs(whippetDataSet, jncCoords, type = NA)
```

Arguments

`whippetDataSet`	whippetDataSet generated from `readWhippetDataSet()`
`jncCoords`	GRanges object with Whippet junctions. Generated by readWhippetJNCfiles()
`type`	type of Whippet event (AA/AD/AF/AL). Note only one event type should be processed at a time.

Value

GRanges object with alternative junctions. Each event should have a set of X (for which the psi measurement is reported) junctions, and alternative Y junctions.

Author(s)

Beth Signal

See Also

Other whippet splicing isoform creation: addIntronInTranscript, findExonContainingTranscripts, findIntronContainingTranscripts, replaceJunction, skipExonInTranscript

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
transcripts <- gtf[gtf$type=="transcript"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.altAce <- filterWhippetEvents(wds, eventTypes="AA")
jncPairs.altAce <- findJunctionPairs(wds.altAce, type="AA")

wds.altDon <- filterWhippetEvents(wds, eventTypes="AD")
jncPairs.altDon <- findJunctionPairs(wds.altDon, type="AD")

wds.altFirst <- filterWhippetEvents(wds, eventTypes="AF", psiDelta=0.2)
jncPairs.altFirst <- findJunctionPairs(wds.altFirst, type="AF")

wds.altLast <- filterWhippetEvents(wds, eventTypes="AL", psiDelta=0.2)
jncPairs.altLast <- findJunctionPairs(wds.altLast, type="AL")
```

---

formatWhippetEvents *Format Whippet co-ordinates as a GRanges object*

Description

Format Whippet co-ordinates as a GRanges object

Usage

```
formatWhippetEvents(whippet)
```

Arguments

whippet	data.frame containing event location information. May be generated by readWhippetDIFFfiles()

Value

GRanges object with events

Author(s)

Beth Signal

See Also

Other whippet data processing: coordinates, diffSplicingResults, filterWhippetEvents, junctions, readCounts, readWhippetDIFFfiles, readWhippetDataSet, readWhippetJNCfiles, readWhippetPSIfiles, whippetTranscriptChangeSummary

Examples

```
whippetFiles <- list.files(system.file("extdata","whippet/",
package = "GeneStructureTools"), full.names = TRUE)
diffFiles <- whippetFiles[grep(".diff", whippetFiles)]
whippetDiffSplice <- readWhippetDIFFfiles(diffFiles)
whippetCoords <- formatWhippetEvents(whippetDiffSplice)
```

---

getOrfs *Get open reading frames for transcripts*

Description

Get open reading frames for transcripts

Usage

```
getOrfs(transcripts, BSgenome = NULL, returnLongestOnly = TRUE,
  allFrames = FALSE, longest = 1, exportFasta = FALSE, fastaFile = NULL,
  uORFs = FALSE)
```

Arguments

transcripts	GRanges object with ONLY exon annotations (no gene, transcript, CDS etc.) with all transcripts for orf retrevial
BSgenome	BSgenome object
returnLongestOnly	only return longest ORF?
allFrames	return longest ORF for all 3 frames?
longest	return x longest ORFs (regardless of frames)
exportFasta	export a .fa.gz file with nucleotide sequences for each transcript?
fastaFile	file name for .fa.gz export
uORFs	get uORF summaries?

Value

data.frame with longest orf details

Author(s)

Beth Signal

See Also

Other ORF annotation: getUOrfs, maxLocation, orfSimilarity

Examples

```
gtf <- rtracklayer::import(system.file("extdata", "example_gtf.gtf",
package="GeneStructureTools"))
transcript <- gtf[gtf$type=="exon" & gtf$gene_name=="Neurl1a"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
# longest ORF for each transcripts
orfs <- getOrfs(transcript, BSgenome = g, returnLongestOnly = TRUE)
# longest ORF in all 3 frames for each transcript
orfs <- getOrfs(transcript, BSgenome = g, allFrames = TRUE)
# longest 3 ORFS in eacht transcript
orfs <- getOrfs(transcript, BSgenome = g, returnLongestOnly = FALSE, longest=3)
```

---

getUOrfs *Get upstream open reading frames for transcripts with annotated main ORFs*

Description

Get upstream open reading frames for transcripts with annotated main ORFs

Usage

```
getUOrfs(transcripts, BSgenome = NULL, orfs, findExonB = FALSE)
```

Arguments

transcripts	GRanges object with ONLY exon annotations (no gene, transcript, CDS etc.) with all transcripts for orf retrevial
BSgenome	BSgenome object
orfs	orf annotation for the transcripts object. Generated by getOrfs(transcripts, ...)
findExonB	find the distance to and exon number of the downstream (B) junction?

Value

data.frame with all upstream ORF details.

Author(s)

Beth Signal

See Also

Other ORF annotation: getOrfs, maxLocation, orfSimilarity

Examples

```
gtf <- rtracklayer::import(system.file("extdata", "example_gtf.gtf",
package="GeneStructureTools"))
transcript <- gtf[gtf$type=="exon" & gtf$gene_name=="Neurl1a"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
# longest ORF for each transcripts
orfs <- getOrfs(transcript, BSgenome = g, returnLongestOnly = FALSE)
uORFS <- getUOrfs(transcript, BSgenome = g, orfs = orfs, findExonB = TRUE)
```

---

junctions *Method junctions*

Description

Method junctions

Usage

```
junctions(whippetDataSet)

## S4 method for signature 'whippetDataSet'
junctions(whippetDataSet)
```

Arguments

`whippetDataSet`	whippetDataSet generated from `readWhippetDataSet()`

Value

junctions GRanges object (originally from a whippet .jnc file)

See Also

Other whippet data processing: `coordinates`, `diffSplicingResults`, `filterWhippetEvents`, `formatWhippetEvents`, `readCounts`, `readWhippetDIFFfiles`, `readWhippetDataSet`, `readWhippetJNCfiles`, `readWhippetPSIfiles`, `whippetTranscriptChangeSummary`

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)

junctions <- junctions(wds)
```

`leafcutterTranscriptChangeSummary` *Compare open reading frames for whippet differentially spliced events*

Description

Compare open reading frames for whippet differentially spliced events

Usage

```
leafcutterTranscriptChangeSummary(significantEvents,
  combineGeneEvents = FALSE, exons, BSgenome, NMD = FALSE,
  showProgressBar = TRUE, exportGTF = NULL)
```

Arguments

`significantEvents`	data.frame containing information from the per_intron_results.tab file output from leafcutter.
`combineGeneEvents`	combine clusters occuring in the same gene? Currently not reccomended.
`exons`	GRanges gtf annotation of exons
`BSgenome`	BSGenome object containing the genome for the species analysed
`NMD`	Use NMD predictions? (Note: notNMD must be installed to use this feature)
`showProgressBar`	show a progress bar of alternative isoform generation?
`exportGTF`	file name to export alternative isoform GTFs (default=NULL)

Value

data.frame containing signficant whippet diff data and ORF change summaries

Author(s)

Beth Signal

Examples

```
leafcutterFiles <- list.files(system.file("extdata","leafcutter/",
package = "GeneStructureTools"), full.names = TRUE)
leafcutterIntrons <- read.delim(leafcutterFiles[
grep("intron_results", leafcutterFiles)],stringsAsFactors=FALSE)
gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
leafcutterTranscriptChangeSummary(significantEvents = leafcutterIntrons,
exons=exons,BSgenome = g,NMD=FALSE)
```

makeGeneModel *Convert GRanges gene model to data.frame for visualisation with Gviz*

Description

Convert GRanges gene model to data.frame for visualisation with Gviz

Usage

```
makeGeneModel(transcript)
```

Arguments

transcript	GRanges of gene model to be visualised

Value

data.frame of a gene model for visualisation

Author(s)

Beth Signal

See Also

Other Gviz gene structure visualisation: annotateGeneModel

Examples

```
gtf <- rtracklayer::import(system.file("extdata", "example_gtf.gtf",
package="GeneStructureTools"))
transcript <- gtf[gtf$type=="exon" & gtf$gene_name=="Neurl1a"]
geneModel <- makeGeneModel(transcript)
```

maxLocation *Find the largest distance between two vectors of numbers Helper function for get_orfs*

Description

Find the largest distance between two vectors of numbers Helper function for get_orfs

Usage

```
maxLocation(startSite, stopSite, longest = 1)
```

Arguments

startSite	vector of start sites - i.e Met amino acid positions
stopSite	vector of stop sites - i.e Stop (*) amino acid positions
longest	which pair to return (1 = longest pair, 2= 2nd longest pair etc.)

Value

sequential start site and end site with the greatest difference

Author(s)

Beth Signal

See Also

Other ORF annotation: `getOrfs`, `getUOrfs`, `orfSimilarity`

Examples

```
starts <- c(1,10,15,25)
stops <- c(4,16,50,55)
# longest start site = 25, longest stop site = 50
maxLocation(starts, stops, longest = 1)
starts <- c(1,10,15,25)
stops <- c(4,14,50,55)
# longest start site = 15, longest stop site = 50
maxLocation(starts, stops, longest = 1)
# 2nd longest start site = 10, 2nd longest stop site = 14
maxLocation(starts, stops, longest = 2)
```

---

orfDiff *Evaluate changes to ORFs caused by alternative splicing*

Description

Evaluate changes to ORFs caused by alternative splicing

Usage

```
orfDiff(orfsX, orfsY, filterNMD = TRUE, geneSimilarity = TRUE,
  compareUTR = TRUE, compareBy = "gene", allORFs = NULL)
```

Arguments

`orfsX`	orf information for 'normal' transcripts. Generated by getOrfs()
`orfsY`	orf information for 'alternative' transcripts. Generated by getOrfs()
`filterNMD`	filter orf information for transcripts not targeted by nmd first?
`geneSimilarity`	compare orf to all orfs in gene?
`compareUTR`	compare UTRs?
`compareBy`	compare by 'transcript' isoforms or by 'gene' groups
`allORFs`	orf information for all transcripts for novel sequence comparisons. Generated by getOrfs()

Value

data.frame with orf changes

Author(s)

Beth Signal

See Also

Other transcript isoform comparisons: attrChangeAltSpliced, transcriptChangeSummary

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
transcripts <- gtf[gtf$type=="transcript"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

orfsProteinCoding <- getOrfs(exons[exons$gene_name=="Prex2" &
exons$transcript_type=="protein_coding"], BSgenome = g)
orfsNMD <- getOrfs(exons[exons$gene_name=="Prex2" &
exons$transcript_type=="nonsense_mediated_decay"], BSgenome = g)
orfDiff(orfsProteinCoding, orfsNMD, filterNMD=FALSE)

wds.exonSkip <- filterWhippetEvents(wds, eventTypes="CE",psiDelta = 0.2)
exons.exonSkip <- findExonContainingTranscripts(wds.exonSkip, exons,
variableWidth=0, findIntrons=FALSE, transcripts)
ExonSkippingTranscripts <- skipExonInTranscript(exons.exonSkip, exons, whippetDataSet=wds.exonSkip)

orfsSkipped <- getOrfs(ExonSkippingTranscripts[ExonSkippingTranscripts$set=="skipped_exon"],
BSgenome = g)
orfsIncluded <- getOrfs(ExonSkippingTranscripts[ExonSkippingTranscripts$set=="included_exon"],
BSgenome = g)
orfDiff(orfsSkipped, orfsIncluded, filterNMD=FALSE)
```

---

orfSimilarity *calculate percentage of orfB contained in orfA*

Description

calculate percentage of orfB contained in orfA

Usage

```
orfSimilarity(orfA, orfB, substitutionCost = 100)
```

Arguments

orfA	character string of ORF amino acid sequence
orfB	character string of ORF amino acid sequence
substitutionCost	cost for substitutions in ORF sequences. Set to 1 if substitutions should be weighted equally to insertions and deletions.

Value

percentage of orfB contained in orfA

Author(s)

Beth Signal

See Also

Other ORF annotation: getOrfs, getUOrfs, maxLocation

Examples

```
orfSimilarity("MFGLDIYAGTRSSFRQFSLT","MFGLDIYAGTRSSFRQFSLT")
orfSimilarity("MFGLDIYAGTRSSFRQFSLT","MFGLDIYAFRQFSLT")
orfSimilarity("MFGLDIYAFRQFSLT","MFGLDIYAGTRSSFRQFSLT")
orfSimilarity("MFGLDIYAGTRXXFRQFSLT","MFGLDIYAGTRSSFRQFSLT")
orfSimilarity("MFGLDIYAGTRXXFSLT","MFGLDIYAGTRSSFRQFSLT", 1)
```

---

overlapTypes *Annotate introns and exonic parts by overlaping exon biotype*

Description

Annotate introns and exonic parts by overlaping exon biotype

Usage

```
overlapTypes(queryCoords, gtf, set = c("from", "to", "overlap"))
```

Arguments

queryCoords	GRanges object of the query regions
gtf	GRanges object of the GTF annotated with exon biotypes - i.e. exon, CDS, UTR
set	which overlapping set of exon biotypes to return - to, from, and/or overlap

Value

overlaping types in a data.frame

Author(s)

Beth Signal

Examples

```
gtfFile <- system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools")
DEXSeqGtfFile <- system.file("extdata","gencode.vM14.dexseq.gtf",
package = "GeneStructureTools")

gtf <- rtracklayer::import(gtfFile)
gtf <- UTR2UTR53(gtf)
DEXSeqGtf <- rtracklayer::import(DEXSeqGtfFile)

overlapTypes(DEXSeqGtf[1:10], gtf)
```

---

readCounts *Method readCounts*

Description

Method readCounts

Usage

```
readCounts(whippetDataSet)

## S4 method for signature 'whippetDataSet'
readCounts(whippetDataSet)
```

Arguments

whippetDataSet	whippetDataSet generated from readWhippetDataSet()

Value

whippet read count data.frame (originally from a whippet .psi file)

See Also

Other whippet data processing: coordinates, diffSplicingResults, filterWhippetEvents, formatWhippetEvents, junctions, readWhippetDIFFfiles, readWhippetDataSet, readWhippetJNCfiles, readWhippetPSIfiles, whippetTranscriptChangeSummary

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)

readCounts <- readCounts(wds)
```

readWhippetDataSet *Import whippet results files as a whippetDataSet*

Description

Import whippet results files as a whippetDataSet

Usage

```
readWhippetDataSet(filePath = ".")
```

Arguments

filePath	path to whippet output files

Value

whippetDataSet

Author(s)

Beth Signal

See Also

Other whippet data processing: coordinates, diffSplicingResults, filterWhippetEvents, formatWhippetEvents, junctions, readCounts, readWhippetDIFFfiles, readWhippetJNCfiles, readWhippetPSIfiles, whippetTranscriptChangeSummary

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
```

readWhippetDIFFfiles *Read in a list of whippet .diff.gz files and format as a data.frame*

Description

Read in a list of whippet .diff.gz files and format as a data.frame

Usage

```
readWhippetDIFFfiles(files)
```

Arguments

files	vector of *.diff.gz file names

Value

data.frame with junction counts for all files

Author(s)

Beth Signal

See Also

Other whippet data processing: coordinates, diffSplicingResults, filterWhippetEvents, formatWhippetEvents, junctions, readCounts, readWhippetDataSet, readWhippetJNCfiles, readWhippetPSIfiles, whippetTranscriptChangeSummary

Examples

```
whippetFiles <- list.files(system.file("extdata","whippet/",
package = "GeneStructureTools"), full.names = TRUE)
diffFiles <- whippetFiles[grep(".diff", whippetFiles)]
whippetDiffSplice <- readWhippetDIFFfiles(diffFiles)
```

---

readWhippetJNCfiles *Read in a list of whippet .jnc.gz files and format as a GRanges object*

Description

Read in a list of whippet .jnc.gz files and format as a GRanges object

Usage

```
readWhippetJNCfiles(files)
```

Arguments

files	vector of *.jnc.gz file names

Value

GRanges object with junctions

Author(s)

Beth Signal

See Also

Other whippet data processing: coordinates, diffSplicingResults, filterWhippetEvents, formatWhippetEvents, junctions, readCounts, readWhippetDIFFfiles, readWhippetDataSet, readWhippetPSIfiles, whippetTranscriptChangeSummary

Examples

```
whippetFiles <- list.files(system.file("extdata","whippet/",
package = "GeneStructureTools"), full.names = TRUE)
jncFiles <- whippetFiles[grep(".jnc", whippetFiles)]
whippetJNC <- readWhippetJNCfiles(jncFiles)
```

`readWhippetPSIfiles` *Read in a list of whippet .psi.gz files and format as a data.frame*

Description

Read in a list of whippet .psi.gz files and format as a data.frame

Usage

```
readWhippetPSIfiles(files, attribute = "Total_Reads", maxNA = NA)
```

Arguments

`files`	vector of *.psi.gz file names
`attribute`	which attribute from the PSI files to use (Total_Reads, Psi, CI_width)
`maxNA`	maximum number of NA values allowed before a site is removed

Value

data.frame with junction counts for all files

Author(s)

Beth Signal

See Also

Other whippet data processing: `coordinates`, `diffSplicingResults`, `filterWhippetEvents`, `formatWhippetEvents`, `junctions`, `readCounts`, `readWhippetDIFFfiles`, `readWhippetDataSet`, `readWhippetJNCfiles`, `whippetTranscriptChangeSummary`

Examples

```
whippetFiles <- list.files(system.file("extdata","whippet/",
package = "GeneStructureTools"), full.names = TRUE)
psiFiles <- whippetFiles[grep(".psi", whippetFiles)]
whippetPSI <- readWhippetPSIfiles(psiFiles)
```

`removeDuplicateTranscripts` *Remove transcript duplicates*

Description

Removes Structural duplicates of transcripts in a GRanges object Note that duplicates must have different transcript ids.

Usage

```
removeDuplicateTranscripts(transcripts)
```

Arguments

`transcripts`	GRanges object with transcript structures in exon form

Value

GRanges object with unique transcript structures in exon form

Author(s)

Beth Signal

See Also

Other gtf manipulation: `UTR2UTR53`, `addBroadTypes`, `exonsToTranscripts`, `filterGtfOverlap`, `removeSameExon`, `reorderExonNumbers`

Examples

```
gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
exons.altName <- exons
exons.altName$transcript_id <- paste(exons.altName$transcript_id, "duplicated", sep="_")
exons.duplicated <- c(exons, exons.altName)
length(exons.duplicated)
exons.deduplicated <- removeDuplicateTranscripts(exons.duplicated)
length(exons.deduplicated)
```

removeSameExon *Remove exon duplicates*

Description

Removes structural duplicates of exons in a GRanges object

Usage

```
removeSameExon(exons)
```

Arguments

exons	GRanges object with exons

Value

GRanges object with unique exons

Author(s)

Beth Signal

See Also

Other gtf manipulation: UTR2UTR53, addBroadTypes, exonsToTranscripts, filterGtfOverlap, removeDuplicateTranscripts, reorderExonNumbers

Examples

```
gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
exons.duplicated <- c(exons[1:4], exons[1:4])
length(exons.duplicated)
exons.deduplicated <- removeSameExon(exons.duplicated)
length(exons.deduplicated)
```

removeVersion *Remove version number from ensembl gene/transcript ids*

Description

Remove version number from ensembl gene/transcript ids

Usage

```
removeVersion(ids)
```

Arguments

ids	vector of ensembl ids

Value

vector of ensembl ids without the version number

Author(s)

Beth Signal

Examples

```
removeVersion("ENSMUSG00000001017.15")
```

`reorderExonNumbers` *Reorder the exon numbers in a gtf annotation*

Description

Reorder the exon numbers in a gtf annotation

Usage

```
reorderExonNumbers(exons, by = "transcript_id")
```

Arguments

exons	GRanges object made from a GTF with ONLY exon annotations (no gene, transcript, CDS etc.)
by	what column are the transcripts grouped by?

Value

The same input GRanges, but with exon numbers reordered.

Author(s)

Beth Signal

See Also

Other gtf manipulation: `UTR2UTR53`, `addBroadTypes`, `exonsToTranscripts`, `filterGtfOverlap`, `removeDuplicateTranscripts`, `removeSameExon`

Examples

```
gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
exons <- reorderExonNumbers(exons)
```

`replaceJunction` *Find transcripts containing/overlapping junctions and replace them with alternative junctions*

Description

Find transcripts containing/overlapping junctions and replace them with alternative junctions

Usage

```
replaceJunction(whippetDataSet, junctionPairs, exons, type = NA)
```

Arguments

`whippetDataSet`	whippetDataSet generated from `readWhippetDataSet()`
`junctionPairs`	GRanges object with alternative Whippet junctions. Generated by findJunctionPairs()
`exons`	GRanges object made from a GTF containing exon coordinates
`type`	type of Whippet event (AA/AD/AF/AL). Note only one event type should be processed at a time.

Value

GRanges object with transcripts containing alternative junctions.

Author(s)

Beth Signal

See Also

Other whippet splicing isoform creation: `addIntronInTranscript`, `findExonContainingTranscripts`, `findIntronContainingTranscripts`, `findJunctionPairs`, `skipExonInTranscript`

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
transcripts <- gtf[gtf$type=="transcript"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.altAce <- filterWhippetEvents(wds, eventTypes="AA")
jncPairs.altAce <- findJunctionPairs(wds.altAce, type="AA")
transcripts.altAce <- replaceJunction(wds.altAce, jncPairs.altAce, exons, type="AA")

wds.altDon <- filterWhippetEvents(wds, eventTypes="AD")
```

```
jncPairs.altDon <- findJunctionPairs(wds.altDon, type="AD")
transcripts.altDon <- replaceJunction(wds.altDon, jncPairs.altDon, exons, type="AD")

wds.altFirst <- filterWhippetEvents(wds, eventTypes="AF", psiDelta=0.2)
jncPairs.altFirst <- findJunctionPairs(wds.altFirst, type="AF")
transcripts.altFirst <- replaceJunction(wds.altFirst, jncPairs.altFirst, exons, type="AF")

wds.altLast <- filterWhippetEvents(wds, eventTypes="AL", psiDelta=0.2)
jncPairs.altLast <- findJunctionPairs(wds.altLast, type="AL")
transcripts.altLast <- replaceJunction(wds.altLast, jncPairs.altLast, exons, type="AL")
```

`skipExonInTranscript` *Remove and include a skipped exon from the transcripts it overlaps*

Description

Remove and include a skipped exon from the transcripts it overlaps

Usage

```
skipExonInTranscript(skippedExons, exons, glueExons = TRUE,
  whippetDataSet = NULL, match = "exact")
```

Arguments

`skippedExons`	data.frame generataed by findExonContainingTranscripts()
`exons`	GRanges object made from a GTF with ONLY exon annotations (no gene, transcript, CDS etc.)
`glueExons`	Join together exons that are not seperated by exons?
`whippetDataSet`	whippetDataSet generated from `readWhippetDataSet()`
`match`	what type of match replacement should be done? exact: exact matches to the skipped event only, also removes any intron overlaps skip: keep non-exact exon match coordinates in included sets, and skip them in skipped sets replace: replace non-exact exon match coordinates with event coordinates in included sets, and skip them in skipped sets

Value

GRanges with transcripts skipping exons

Author(s)

Beth Signal

See Also

Other whippet splicing isoform creation: `addIntronInTranscript`, `findExonContainingTranscripts`, `findIntronContainingTranscripts`, `findJunctionPairs`, `replaceJunction`

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
transcripts <- gtf[gtf$type=="transcript"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.exonSkip <- filterWhippetEvents(wds, eventTypes="CE",psiDelta = 0.2)
exons.exonSkip <- findExonContainingTranscripts(wds.exonSkip, exons,
variableWidth=0, findIntrons=FALSE, transcripts)
ExonSkippingTranscripts <- skipExonInTranscript(exons.exonSkip, exons, whippetDataSet=wds.exonSkip)

exonFromGRanges <- exons[exons$exon_id == "ENSMUSE00001271768.1"]
exons.exonSkip <- findExonContainingTranscripts(exonFromGRanges, exons,
variableWidth=0, findIntrons=FALSE, transcripts)
ExonSkippingTranscripts <- skipExonInTranscript(exons.exonSkip, exons, match="skip")
```

---

summariseExonTypes *Summarise exon biotypes to broader categories*

Description

Summarise exon biotypes to broader categories

Usage

```
summariseExonTypes(types)
```

Arguments

types	vector of exon biotypes

Value

vector of broader exon biotypes

Author(s)

Beth Signal

See Also

Other DEXSeq processing methods: DEXSeqIdsToGeneIds, findDEXexonType

Examples

```
gtfFile <- system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools")
DEXSeqGtfFile <- system.file("extdata","gencode.vM14.dexseq.gtf",
package = "GeneStructureTools")

gtf <- rtracklayer::import(gtfFile)
gtf <- UTR2UTR53(gtf)
DEXSeqGtf <- rtracklayer::import(DEXSeqGtfFile)

findDEXexonType("ENSMUSG00000032366.15:E028", DEXSeqGtf, gtf)

DEXSeqResultsFile <- system.file("extdata","dexseq_results_significant.txt",
package = "GeneStructureTools")
DEXSeqResults <- read.table(DEXSeqResultsFile, sep="\t")

types <- findDEXexonType(rownames(DEXSeqResults), DEXSeqGtf, gtf)
summarisedTypes <- summariseExonTypes(types)
table(types, summarisedTypes)
```

---

transcriptChangeSummary

Compare open reading frames for two sets of paired transcripts

---

Description

Compare open reading frames for two sets of paired transcripts

Usage

```
transcriptChangeSummary(transcriptsX, transcriptsY, BSgenome, exons,
  NMD = FALSE, NMDModel = NULL, compareBy = "gene",
  orfPrediction = "allFrames", compareToGene = FALSE,
  whippetDataSet = NULL, exportGTF = NULL)
```

Arguments

transcriptsX	GRanges object with exon annotations for all transcripts to be compared for the 'normal' condition
transcriptsY	GRanges object with exon annotations for all transcripts to be compared for the 'alternative' condition
BSgenome	BSGenome object containing the genome for the species analysed
exons	GRanges object made from a GTF containing exon coordinates
NMD	Use NMD predictions? (Note: notNMD must be installed to use this feature)
NMDModel	Use the "base" or "lncRNA" NMD model?
compareBy	compare isoforms by 'transcript' id, or aggregate all changes occuring by 'gene'
orfPrediction	What type of orf predictions to return. default= "allFrames"
compareToGene	compare alternative isoforms to all normal gene isoforms (in exons)
whippetDataSet	whippetDataSet generated from readWhippetDataSet() Use if PSI directionality should be taken into account when comparing isoforms.
exportGTF	file name to export alternative isoform GTFs (default=NULL)

Value

Summarised ORF changes data.frame

Author(s)

Beth Signal

See Also

Other transcript isoform comparisons: attrChangeAltSpliced, orfDiff

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
exons <- gtf[gtf$type=="exon"]
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10

wds.exonSkip <- filterWhippetEvents(wds, eventTypes="CE",psiDelta = 0.2)

exons.exonSkip <- findExonContainingTranscripts(wds.exonSkip, exons,
variableWidth=0, findIntrons=FALSE, transcripts)
ExonSkippingTranscripts <- skipExonInTranscript(exons.exonSkip, exons, whippetDataSet=wds.exonSkip)
transcriptChangeSummary(ExonSkippingTranscripts[ExonSkippingTranscripts$set=="included_exon"],
ExonSkippingTranscripts[ExonSkippingTranscripts$set=="skipped_exon"],
BSgenome=g,exons)
```

UTR2UTR53 *Annotate UTRs from Gencode GTF as 5' or 3'*

Description

Annotate UTRs from Gencode GTF as 5' or 3'

Usage

```
UTR2UTR53(gtf)
```

Arguments

gtf	GRanges object of the GTF

Value

gtf annotation GRanges object

Author(s)

Beth Signal

See Also

Other gtf manipulation: addBroadTypes, exonsToTranscripts, filterGtfOverlap, removeDuplicateTranscripts, removeSameExon, reorderExonNumbers

Examples

```
gtfFile <- system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools")
gtf <- rtracklayer::import(gtfFile)
gtf <- UTR2UTR53(gtf)
table(gtf$type)
```

---

whippetDataSet-class *Class whippetDataSet*

Description

Class whippetDataSet contains information read from whippet output files

---

whippetTranscriptChangeSummary *Compare open reading frames for whippet differentially spliced events*

Description

Compare open reading frames for whippet differentially spliced events

Usage

```
whippetTranscriptChangeSummary(whippetDataSet, gtf.all = NULL, BSgenome,
  eventTypes = "all", exons = NULL, transcripts = NULL, NMD = FALSE,
  exportGTF = NULL)
```

Arguments

whippetDataSet	whippetDataSet generated from readWhippetDataSet()
gtf.all	GRanges gtf annotation (can be used instead of specifying exons and transcripts)
BSgenome	BSGenome object containing the genome for the species analysed
eventTypes	which event type to filter for? default = "all"
exons	GRanges gtf annotation of exons
transcripts	GRanges gtf annotation of transcripts
NMD	Use NMD predictions? (Note: notNMD must be installed to use this feature)
exportGTF	file name to export alternative isoform GTFs (default=NULL)

Value

data.frame containing signficant whippet diff data and ORF change summaries

Author(s)

Beth Signal

See Also

Other whippet data processing: coordinates, diffSplicingResults, filterWhippetEvents, formatWhippetEvents, junctions, readCounts, readWhippetDIFFfiles, readWhippetDataSet, readWhippetJNCfiles, readWhippetPSIfiles

Examples

```
whippetFiles <- system.file("extdata","whippet/",
package = "GeneStructureTools")
wds <- readWhippetDataSet(whippetFiles)
wds <- filterWhippetEvents(wds)

gtf <- rtracklayer::import(system.file("extdata","example_gtf.gtf",
package = "GeneStructureTools"))
g <- BSgenome.Mmusculus.UCSC.mm10::BSgenome.Mmusculus.UCSC.mm10
whippetTranscriptChangeSummary(wds, gtf.all=gtf,BSgenome = g)
```

Index