Package ‘IntEREst’

April 12, 2022

Title Intron-Exon Retention Estimator

Version 1.18.0

Date 2016-12-15

Author Ali Oghabian <Ali.Oghabian@Helsinki.Fi>, Dario Greco <dario.greco@helsinki.fi>, Mikko Frilander <Mikko.Frilander@helsinki.fi>

Maintainer Ali Oghabian <Ali.Oghabian@Helsinki.Fi>, Mikko Frilander <Mikko.Frilander@helsinki.fi>

Description This package performs Intron-Exon Retention analysis on RNA-seq data (.bam files).

Depends R (>= 3.4), GenomicRanges, Rsamtools, SummarizedExperiment, edgeR, S4Vectors

Imports seqLogo, Biostrings, GenomicFeatures (>= 1.39.4), IRanges, seqinr, graphics, grDevices, stats, utils, grid, methods, DBI, RMySQL, GenomicAlignments, BiocParallel, BiocGenerics, DEXSeq, DESeq2

Suggests clinfun, knitr, rmarkdown, BSgenome.Hsapiens.UCSC.hg19

VignetteBuilder knitr

LazyData true

biocViews Software, AlternativeSplicing, Coverage, DifferentialSplicing, Sequencing, RNASeq, Alignment, Normalization, DifferentialExpression, ImmunoOncology

License GPL-2

NeedsCompilation no

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

git_branch RELEASE_3_14

git_last_commit 0a4f63a

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`IntEREst-package` *IntEREst*

Description

Intron/Exon retention estimator quantifies and normalizes Intron retention and Exon junction read levels by analyzing mapped reads (.bam) files.

Details

Package:	IntEREst
Type:	Package
Version:	1.0
Date:	2015-11-18
License: GPL-2	

To run the pipeline use functions `interest()` or `interest.sequential()`, i.e. wrapper functions that run all the necessary functions.

Author(s)

Ali Oghabian <Ali.Oghabian@Helsinki.Fi>, Dario Greco <dario.greco@helsinki.fi>, Mikko Frilander <Mikko.Frilander@helsinki.fi>

Maintainer: Ali Oghabian <Ali.Oghabian@Helsinki.Fi>, Mikko Frilander <Mikko.Frilander@helsinki.fi>

`addAnnotation` *Adding sample annotations to a* `SummarizedExperiment` *object*

Description

Adds a new sample annotation to the `SummarizedExperiment` object. In other words it adds and column with sample annotations to the colData of the `SummarizedExperiment` object.

Usage

```
addAnnotation(x, sampleAnnotationType, sampleAnnotation)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`sampleAnnotationType`	The name of the new column to be added to the `colData` table of `SummarizedExperiment` object.

`sampleAnnotation`	Vector with the same length as the row-size of the `colData` attribute of the `SummarizedExperiment` object, which includes the sample annotations.

Value

An InterestResult object.

Author(s)

Ali Oghabian

See Also

`getAnnotation`

Examples

```
# Check the annotation table of mdsChr22Obj data
getAnnotation(mdsChr22Obj)

# Add a new sample annotation
newMdsChr22Obj <- addAnnotation(x=mdsChr22Obj,
sampleAnnotationType="sample_number",
sampleAnnotation=1:16
)

# Retrieve annotations of the new object
getAnnotation(newMdsChr22Obj)
```

annotateU12 *Annotate the U12 (and U2) type introns*

Description

Receives coordinates, a reference genome and PWMs of splice site of U12 and U2 type introns, and returns a data.frame with 2 columns. The first column shows wheather the corresponding sequences matches U12, U2 or both (U12/U2) consensus sequences (based on their score when fitting the PWMs). The second column shows whether the match is on positive strand or negative when fitting the PWMs to the sequences.

Usage

```
annotateU12(pwmU12U2=c(), pwmSsIndex=c(), referenceChr, referenceBegin,
referenceEnd, referenceIntronExon, intronExon='intron',
matchWindowRelativeUpstreamPos=c() , matchWindowRelativeDownstreamPos=c(),
minMatchScore='80%', refGenome='', setNaAs='U2', annotateU12Subtype=TRUE,
includeMatchScores=FALSE, ignoreHybrid=TRUE, filterReference)
```

Arguments

`pwmU12U2`	A list containing position weight matrices of (in order): Donor site, branch point, and acceptor site of U12-type introns, and donor site and acceptor site of U2-type introns. If not provided, the information related to `pwmU12db` data is used.
`pwmSsIndex`	A list (or vector) that contains the column number in each element of `pwmU12U2` that represents the 5' or 3' Splice Site; The order should be equivalent to the `pwmU12U2`. If not provided the information from `pwmU12db` data is used, i.e. `pwmSsIndex=list(indexDonU12=1,indexBpU12=1,indexAccU12=3,indexDonU2=1,indexAccU2=3)`
`referenceChr`	Chromosome names of the references (e.g. introns).
`referenceBegin`	A vector that corresponds to the begin coordinates of the reference (e.g. introns).
`referenceEnd`	A vector that corresponds to the end coordinates of the reference (e.g. introns). `referenceEnd` should be greater than or equal to `referenceBegin`.
`referenceIntronExon`	A vector with the same size as the `referenceChr`, `referenceBegin` and `referenceEnd` which contains `'intron'` and `'exon'` describing what (either intron or exon) each element of the 3 vectors represents.
`intronExon`	Should be assigned either `'intron'` or `'exon'` or `c('intron','exon')` based on whether match the PWM to the intronic, exonic, or intronic and exonic regions of the reference. By default it seeks matches in intronic regions (intronExon='intron').
`matchWindowRelativeUpstreamPos`	A vector the same size as the `pwmU12U2` (and the same order of donor/acceptor sites' information in `pwmU12U2`) which consists of the upstream distance from the donor/acceptor site from which each PWM should be tested (to see if they match). If not provided, the information from `pwmU12db` data is used i.e. `matchWindowRelativeUpstrean` `c(NA,-29,NA,NA,NA)`.
`matchWindowRelativeDownstreamPos`	A vector the same size as the `pwmU12U2` (and the same order of donor/acceptor sites' information in `pwmU12U2`) which consists of the downstream distance from the donor/acceptor site to which each PWM should be tested (to see if they match). If not provided, the information from `pwmU12db` data is used i.e. `matchWindowRelativeDownstre` `c(NA,-9,NA,NA,NA)`.
`minMatchScore`	Min percentage match score, when scoring matching of a sequence to `pwm`. Different score thresholds could also be defined for the various sites (U12/U2 donors, the U12 branch point and U12/U2 acceptors); A vector with 5 elements can be assigned which each shows the match score to use for each PWM in `pwmU12U2`.
`refGenome`	The reference genome; Object of class BSgenome. Use `available.genome()` from the `BSgenome` package to see the available genomes. `DNAStringSet` objects (from `Biostrings` package) and fasta files are also accepted as input.
`setNaAs`	Defines that if reference (e.g. intron) did not match any of U12 or U2 type introns based on the scores obtained from PWM what should the function return. If an intron was not proven to be U12 or U2 based on PWM scores it can be considered as U2-type since the U12 type introns constitute for about 1% of introns in human genome and they are muxh more conserved than the U2 type

introns, hence the default is 'U2'; otherwise it is also possible to set it as NA or nan or 'U12/U2'.

`annotateU12Subtype`
: Whether annotate the subtypes of the U12 type Introns. The value is `TRUE` by default.

`includeMatchScores`
: If set as TRUE the final data frame result includes the PWM match scores (FALSE by default).

`ignoreHybrid`
: Whether ignore the U12 hybrid subtypes, i.e. GT-AC and AT-AG (TRUE by default).

`filterReference`
: Optional parameter that can be defined either as a GRanges or SummarizedExperiment object. If defined as the latter, the first 3 columns of the rowData must be: chr name, start and end of the coordinates. If the parameter is defined the introns/exon coordinates will be mapped against it and the intron type of all those that do not match will be set as NA.

Value

Data frame containing 3 columns representing (in order): intron type (U12, U2 or none), strand match indicating whether the PWM matches to the sequence (+ strand) or the reverese complement of the sequence (- strand) or none (NA), and the U12 subtype (GT-AG or AT-AC). If includeMatchScores is set as TRUE further columns that include the PWM match scores will also be included.

Author(s)

Ali Oghabian

See Also

`buildSsTypePwms`.

Examples

```
# Improting genome
BSgenome.Hsapiens.UCSC.hg19 <-
BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
#Choosing subset of rows
ind<- 69:94
# Annotate U12 introns with strong U12 donor site, branch point
# and acceptor site from the u12 data in the package
annoU12<-
annotateU12(pwmU12U2=list(pwmU12db[[1]][,11:17],pwmU12db[[2]]
,pwmU12db[[3]][,38:40],pwmU12db[[4]][,11:17],
pwmU12db[[5]][,38:40]),
pwmSsIndex=list(indexDonU12=1, indexBpU12=1, indexAccU12=3,
indexDonU2=1, indexAccU2=3),
referenceChr=u12[ind,'chr'],
referenceBegin=u12[ind,'begin'],
referenceEnd=u12[ind,'end'],
referenceIntronExon=u12[ind,"int_ex"],
intronExon="intron",
matchWindowRelativeUpstreamPos=c(NA,-29,NA,NA,NA),
matchWindowRelativeDownstreamPos=c(NA,-9,NA,NA,NA),
minMatchScore=c(rep(paste(80,"%",sep=""),2), "60%",
paste(80,"%",sep=""), "60%"),
refGenome=BSgenome.Hsapiens.UCSC.hg19,
setNaAs="U2",
annotateU12Subtype=TRUE)

# How many U12 and U2 type introns with strong U12 donor sites,
# acceptor sites (and branch points for U12-type) are there?
table(annoU12[,1])
```

applyOverlap *Apply function over counts*

Description

Runs a function on columns of the counts (assay) of a 'SummarizedExperiment' object (resulted by `interest()`, `interest.sequential()` or `readInterestResults()`) based on the overalp of its exon/intron coordinates with those of another 'SummarizedExperiment' object. The number of the rows and the dimensions of the counts of the result are equal to those of the `subject`. The function is applied on the query based on it's overlap to the subject.

Usage

```
applyOverlap(
query,
subject,
type="any",
replaceValues=FALSE,
intExCol="int_ex",
intronExon="intron",
sujectGeneNamesCol,
repeatsTableToFilter=c(),
scaleFragment=TRUE,
scaleLength=TRUE,
unmapValue=0,
FUN=mean,
...
)
```

Arguments

`query, subject`	SummarizedExperiment objects resulted by `interest()`, `interest.sequential()` or `readInterestResults()` functions.
`type`	The type of overlap. By default it considers any overlap. See findOverlaps-methods for more info.
`replaceValues`	Whether return a 'SummarizedExperiment' object with new counts (resulted by running function) replaced.
`intExCol`	Column name (or number) in the rowData of the objects that represents whether each row of the assay is "intron" or "exon".
`intronExon`	Should be assigned either `'intron'` or `'exon'` or `c('intron','exon')` based on whether match the PWM to the intronic, exonic, or intronic and exonic regions of the reference. By default it seeks matches in intronic regions (intronExon='intron').
`sujectGeneNamesCol`	The column in the row data of the subject that includes the gene names.
`repeatsTableToFilter`	A data.frame table that includes chr,begin and end columns. If defined, all reads mapped to the described regions will be ingnored.
`scaleFragment`	Logical value, indicating whether the retention levels must be scaled by (genewide) fragment levels.
`scaleLength`	Logical value, indicating whether the retention levels must be scaled by length of the introns/exons.
`unmapValue`	The value to assign to unmapped rows (i.e. introns/exons).
`FUN`	The function to apply.
`...`	Other parameter settings from `aggregate()` function.

Value

The returned value is a data frame if `replaceValues` is FALSE and it is SummarizedExperiment if `replaceValues` is TRUE.

Author(s)

Ali Oghabian

See Also

`readInterestResults` `interest` `interest.sequential`

Examples

```
mdsChr22Obj

tmp<- applyOverlap(
query=mdsChr22Obj,
subject=mdsChr22Obj,
type="equal",
replaceValues=FALSE,
intExCol="int_ex",
intronExon="intron",
sujectGeneNamesCol="collapsed_transcripts",
scaleFragment=TRUE,
scaleLength=TRUE,
unmapValue=0,
FUN=head,
n=1
)
```

`attributes` *Extracting values of useful attributes of SummarizedExperiment objects*

Description

Several functions are provided that can extract various attribnutes from an object of class `SummarizedExperiment` generated by IntEREst functions, e.g. `interest()`, `interest`, and `readInterestResults`. It is possible to extract sample annotations using `getAnnotation` function. One can also extract the scaled retention levels of the introns/exons using `scaledRetention()` function. Notes that `colData` and `rowData` methods of `SummarizedExperiment` class can also be used to extract row and column data.

Usage

```
getAnnotation(x)
scaledRetention(x)
```

Arguments

`x`	Object of type `SummarizedExperiment`.

Value

Various data types (data.frame/vector) dependent on the function used. See the "Description" for more information.

Author(s)

Ali Oghabian

See Also

`SummarizedExperiment-class` `addAnnotation` `counts-method` `plot-method`

Examples

```
# Retrieve the sample annotations from mdsChr22Obj
getAnnotation(mdsChr22Obj)
# Retrieving the scaled retention levels from mdsChr22Obj
head(scaledRetention(mdsChr22Obj))

#for row and column data SummarizedExperiment methods can be used
head(rowData(mdsChr22Obj))
colData(mdsChr22Obj)
```

---

boxplot-method *boxplot - method*

Description

boxplot method for `SummarizedExperiment` objects.

Usage

```
## S4 method for signature 'SummarizedExperiment'
boxplot(x, sampleAnnoCol=NA,
intexTypeCol="int_type", intexType=c(), col="white", boxplotNames=c(),
lasNames=3, outline=FALSE, addGrid=FALSE, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment` generated by either `interest()`, `interest.sequential()` or `readInterestResults()`.
`sampleAnnoCol`	Which colummn of `colData` in `x` to consider for plotting.
`intexTypeCol`	Column name (or number) that represents what type of intron/exon each row of `x` assays represents.
`intexType`	A vector of characters describing types of introns/exons to be plotted. They must be elements in the `intexTypeCol` column of the rowData of `x`. rowData of `x` is a dataframe that includes various annotations of the introns/exons.
`col`	Vector showing box colours. It is either of size 1 or the same size as the number of groups to be plotted.
`boxplotNames`	Names to write under boxes. If not defined, as names, it pastes the row (intron/exon) annotation names to the sample group annotations separated by a space " ".
`lasNames`	Orientation of the box names.
`outline`	If outline is TRUE the outlier points are drawn otherwise if FALSE (default) they are not.
`addGrid`	Whether add a grid under the boxplots (FALSE by default).
`...`	Other arguments to pass to the `boxplot()` and `axis` function.

Value

Returns NULL.

Author(s)

Ali Oghabian

See Also

Class: `SummarizedExperiment-class` Method: `counts-method` `plot-method`

Examples

```
#Plotting U12- vs U2-type introns
par(mar=c(8,4,2,1))
boxplot(x=mdsChr22Obj, sampleAnnoCol="type", intexTypeCol="intron_type",
intexType=c("U2", "U12"),
col=rep(c("yellow", "orange"),3),
boxplotNames=c(), lasNames=3, outline=FALSE,
addGrid=TRUE)
```

`buildSsTypePwms` *Building Position Weight Matrices for Splice Sites of U12 and U2 type introns.*

Description

Builds position Weigh Matrices for the donor and acceptor sites of the U12 and U2 type introns, and the branchpoint of the U12 type introns. if `pdfFileSeqLogos` is defined a pdf is also produced that contains the sequence logos of the results. The result is a list that contains PWMs of the splice sites of U12 and U2 dependent introns.

Usage

```
buildSsTypePwms( cexSeqLogo=1, pdfWidth=35, pdfHeight=10, tmpDir="./",
u12dbSpecies="Homo_sapiens",
pwmSource="U12DB",
u12DonorBegin, u12BranchpointBegin, u12AcceptorBegin,
u2DonorBegin, u2AcceptorBegin, u12DonorEnd,
u12BranchpointEnd, u12AcceptorEnd, u2DonorEnd,
u2AcceptorEnd, pasteSites=FALSE,
splicerackSsLinks=list(
U12_AT_AC_donor=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.25",
U12_AT_AC_branchpoint=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.26",
U12_AT_AC_acceptor=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.29",
U12_GT_AG_donor=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.22",
U12_GT_AG_branchpoint=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.27",
U12_GT_AG_acceptor=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.21",
U2_GC_AG_donor="http://katahdin.mssm.edu/splice/out/9606_logo_file.24",
U2_GC_AG_acceptor=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.30",
U2_GT_AG_donor="http://katahdin.mssm.edu/splice/out/9606_logo_file.23",
U2_GT_AG_acceptor=
"http://katahdin.mssm.edu/splice/out/9606_logo_file.28"),
u12dbLink="https://genome.crg.cat/pub/software/u12/u12db_v1_0.sql.gz",
u12dbDbName="u12db", u12dbDropDb=TRUE,  pdfFileSeqLogos="",
removeTempFiles=TRUE, ...)
```

Arguments

`cexSeqLogo`	Font size of sequence logo plots; used only if `pdfFileSeqLogos` is defined.
`pdfWidth, pdfHeight`	The width and height of the graphics region of the pdf in inches. The default values are 35 and 10.
`tmpDir`	Path to directory used for storing temporary files.
`u12dbSpecies`	What species data to use when getting the data from the U12DB database (`pwmSource="U12DB"`).
`pwmSource`	The source used to buildSplice Sites of U12 and U2 type introns the PWM for U12 and U2 dependent introns. Default is `U12DB`; but also accepts `SpliceRack`.
`u12DonorBegin, u12DonorEnd`	Integer values. They correspond to the begin and end point of the donor sequences of U12-type introns to consider (optional).
`u12BranchpointBegin, u12BranchpointEnd`	Integer values. Begin and end points of the branch point sequences of U12-type introns (optional).
`u12AcceptorBegin, u12AcceptorEnd`	Integer values. Begin and end points of the acceptor sequences of U12-type introns (optional).
`u2DonorBegin, u2DonorEnd`	Integer values. Begin and end points of the donor sequences of U2-type introns (optional).
`u2AcceptorBegin, u2AcceptorEnd`	Integer values. Begin and end points of the acceptor sequences of U2-type introns (optional).
`pasteSites`	Logical. If TRUE the donor, branch point and acceptor seqs are pasted before a PWM is built; then the PWMs of each (donor, acceptor and bp) are assigned. If FALSE (default) the PWMs for each is built separately.

`spliceRackSsLinks`	A list (or vector) that contains the SpliceRack URL links to the text files that contain Position Weigh Matrices of the splice sites of U12 and U2 introns. This parameter is used only when pwmSource="SpliceRack". You can get the links to PWM files from this URL (choose logo files with "File" links): `http://katahdin.mssm.edu/splice/splice_matrix.cgi?database=spliceNew`. The links should be defined in the following order: `U12_AT_AC_donor`, `U12_AT_AC_branchpoint`, `U12_AT_AC_acceptor`, `U12_GT_AG_donor`, `U12_GT_AG_branchpoint`, `U12_GT_AG_acceptor`, `U2_GC_AG_donor`, `U2_GC_AG_acceptor`, `U2_GT_AG_donor`, and `U2_GT_AG_acceptor`.
`u12dbLink`	A character string containing the URL for downloading the zipped MySQL dump file of the U12DB. Used when `pwmSource="U12DB"`.
`u12dbDbName`	Name of the database copy of the U12DB that is build locally. Used when `pwmSource="U12DB"`.
`u12dbDropDb`	Drop (or remove) the local copy of the U12DB database at the end of the run. Used when `pwmSource="U12DB"`.
`pdfFileSeqLogos`	Path to PDF file containing the sequence logos of the results. By default it does not produce a file.
`removeTempFiles`	Whether remove temporary files at the end of the run; accepts `TRUE` or `FALSE` values (default is `TRUE`).
`...`	Authorization arguments needed by the DBMS instance. See the manual for `dbConnect` of the `DBI` package for more info.

Value

`pwmDonorU12`	Matrix (with 4 rows represnting A, C, G, T and `n` columns representing the genomic coordinates) representing the Position Weight Matrix of donor site of U12-type introns.
`pwmBpU12`	Position Weight Matrix of branchpoint of U12-type introns.
`pwmAccU12`	Position Weight Matrix of acceptor site of U12-type introns.
`pwmDonU2`	Position Weight Matrix of donor site of U2-type introns.
`pwmAccU2`	Position Weight Matrix of acceptor site of U2-type introns.

Author(s)

Ali Oghabian

See Also

`annotateU12`.

Examples

```
# Time demanding function
## Not run:
#Build temp directory
```

```
tmpDir<- tempdir()


# Creating subdirectory for storing u12db temp files
dir.create(paste(tmpDir, "u12dbTmp", sep="/"))

# Extracting PWMs of Splice Sites of U12 and U2 type introns -
# based on u12db
u12dbPwm<-buildSsTypePwms(
tmpDir=paste(tmpDir, "u12dbTmp", sep="/"),
u12dbSpecies="Homo_sapiens",
resource="U12DB",
u12dbDbName="u12db",
u12dbDropDb=TRUE,
removeTempFiles=TRUE)


# Creating subdirectory for storing SpliceRack temp files
dir.create(paste(tmpDir, "splicerackTmp", sep="/"))

# Extracting PWMs of Splice Sites of U12 and U2 type introns -
# based on SpliceRack
spliceRackPwm<- buildSsTypePwms(
tmpDir= paste(tmpDir, "splicerackTmp", sep="/"),
resource="SpliceRack",
removeTempFiles=TRUE)

## End(Not run)
```

`counts-method` *Counts - method*

Description

Returns the (row) number of reads that are mapped to introns/exons in various samples.

Usage

```
## S4 method for signature 'SummarizedExperiment'
counts(object)
```

Arguments

`object`	Object of type `SummarizedExperiment`.

Value

Returns a numeric matrix.

Author(s)

Ali Oghabian

See Also

Class: SummarizedExperiment-class

Method: plot-method.

Examples

```
#Show contents of a InterestResults object included in IntEREst
head(counts(mdsChr22Obj))

#Make a test InterestResults object
geneId<- paste("gene", c(rep(1,5), rep(2,5), rep(3,5), rep(4,5)),
sep="_")
readCnt1<- sample(1:100, 20)
readCnt2<- sample(1:100, 20)
readCnt3<- sample(1:100, 20)
readCnt4<- sample(1:100, 20)
fpkm1<- readCnt1/(tapply(readCnt1, geneId, sum))[geneId]
fpkm2<- readCnt2/(tapply(readCnt2, geneId, sum))[geneId]
fpkm3<- readCnt3/(tapply(readCnt3, geneId, sum))[geneId]
fpkm4<- readCnt4/(tapply(readCnt4, geneId, sum))[geneId]

# Creating object using test data
interestDat<- data.frame(
int_ex=rep(c(rep(c("exon","intron"),2),"exon"),4),
int_ex_num= rep(c(1,1,2,2,3),4),
gene_id= geneId,
sam1_readCnt=readCnt1,
sam2_readCnt=readCnt2,
sam3_readCnt=readCnt3,
sam4_readCnt=readCnt4,
sam1_fpkm=fpkm1,
sam2_fpkm=fpkm2,
sam3_fpkm=fpkm3,
sam4_fpkm=fpkm4
)
readFreqColIndex<- grep("_readCnt$",colnames(interestDat))
scaledRetentionColIndex<- grep("_fpkm$",colnames(interestDat))

scalRetTmp<- as.matrix(interestDat[ ,scaledRetentionColIndex])
colnames(scalRetTmp)<-gsub("_fpkm$","", colnames(scalRetTmp))

frqTmp<- as.matrix(interestDat[ ,readFreqColIndex])
colnames(frqTmp)<-gsub("_readCnt$","", colnames(frqTmp))


InterestResultObj<- InterestResult(
resultFiles=paste("file",1:4, sep="_"),
rowData= interestDat[ , -c(readFreqColIndex,
scaledRetentionColIndex)],
counts= frqTmp,
scaledRetention= scalRetTmp,
scaleLength=TRUE,
scaleFragment=FALSE,
sampleAnnotation=data.frame(
sampleName=paste("sam",1:4, sep=""),
gender=c("M","M","F","F"),
health=c("healthy","unhealthy","healthy","unhealthy")
, row.names=paste("sam", 1:4, sep="")
)
)

#Show
head(counts(InterestResultObj))
```

`deseqInterest` *DESeq2 analysis for IntEREst object*

Description

Differential intron retention test adapted from the `DESeq2` package.

Usage

```
deseqInterest (x, design, pAdjustMethod = "BH",
sizeFactor=c(), contrast, bpparam, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`design`	Formula specifying the design of the experiment. It must specify an interaction term between variables from column names of `sampleData(x)`.
`pAdjustMethod`	What adjustment method to be sed on the p-values. See p.adjust for more information.
`sizeFactor`	Numeric vector with the same size as the clolumn size of the count matrix in `x`, if defined it will be used for scaling of the count matrix.
`contrast`	Argument speciftying the comparison to extract from `x`. See `results` function in the `DESeq2` package for more information.
`bpparam`	An optional `BiocParallelParam` instance defining the parallel back-end to be used. If not defined the function will run sequentially (on a single computing core).
`...`	Other parameter settings for the results function in the `DESeq2` package.

Value

a `DESeqResults` object.

Author(s)

Ali Oghabian

See Also

exactTestInterest qlfInterest, treatInterest DEXSeqIntEREst

Examples

```
mdsChr22IntObj<- mdsChr22Obj[rowData(mdsChr22Obj)$int_ex=="intron",]
deseqRes<- deseqInterest(x=mdsChr22IntObj,
design=~test_ctrl, contrast=list("test_ctrl_test_vs_ctrl"))

# Number of U12/U2 type significantly differential retained introns in chr22
table(rowData(mdsChr22Obj)[which(deseqRes$padj<.01), "intron_type"])
```

DEXSeqIntEREst *DEXSeq test for IntEREst object*

Description

Genewise differential exon usage or intron retention test adapted from the DEXSeq package.

Usage

```
DEXSeqIntEREst (x, design, reducedModel = ~ sample + intex, fitExpToVar,
intExCol, geneIdCol, bpparam, silent=TRUE,...)
```

Arguments

x	Object of type SummarizedExperiment.
design	Formula specifying the design of the experiment. It must specify an interaction term between a variable from columns of sampleData(x) with one of the 'exon', 'intron' or 'intex' (i.e. intron and exon) variables; based on which of these variables are used (exon, intron , or 'intex') the x will be filtered reletively to include exons, introns , or introns and exons. See DEXSeqDataSet for more information.
reducedModel	The null model formula. By default it is '~ sample + intex'.
fitExpToVar	A variable name contained in the column data (i.e. column names of colData(x)). See DEXSeq for more information.
intExCol	Column name (or number) that represents whether each row is "intron" or "exon" in rowData of x.
geneIdCol	Column name (or number of column) in rowData of x, i.e. SummarizedExperiment object, that represents the gene ID of the introns and exons in x.
bpparam	An optional BiocParallelParam instance defining the parallel back-end to be used.

`silent`	Whether run the DEXSeq function silently (if TRUE) or allow it to print messages at each step (if FALSE).
`...`	Other parameter settings for the `DEXSeqDataSet` function in the `DEXSeq` package.

Details

The `design` and `reduceModel` accept formula that specify the design of the experiment. The formula must describe an interaction between variables from columns of `sampleData(x)` with one of the 'exon', 'intron' or 'intex' (i.e. intron and exon) variables; Based on which of these variables are used (exon, intron , or 'intex') the input object (`x`) will be filtered reletively to include exons, introns , or introns and exons. Hence the number of the rows of the returned value is equal to the number of the rows of the filtered object, i.e. the number of the exons, introns or both based on the `design` formula.

Value

A `DEXSeqResults` object.

Author(s)

Ali Oghabian

See Also

`exactTestInterest`

Examples

```
dexseqExRes<-DEXSeqIntEREst (x=mdsChr22ExObj,
design= ~ sample + exon + test_ctrl:exon,
reducedModel = ~ sample + exon, fitExpToVar="test_ctrl",
intExCol="int_ex", geneIdCol="transcripts_id", silent=TRUE)
head(dexseqExRes)
```

`exactTestInterest` *Exact test*

Description

Compute genewise exact test between two groups of read counts, using the `edgeR` package.

Usage

```
exactTestInterest(x, sampleAnnoCol=c(), sampleAnnotation=c(),
geneIdCol, silent=TRUE, group=c(), rejection.region="doubletail",
big.count=900, prior.count=0.125, disp="common", ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`sampleAnnoCol`	Which colummn of `colData` of `x` to consider for the analysis.
`sampleAnnotation`	A vector of size 2 which cotains values from `colData` of `SummarizedExperiment` object; e.g. if `getAnnotation(x)[,sampleAnnoCol]= c("test","test","ctrl","ctrl",...)` , and the goal is to compare "test" and "ctrl" samples, `sampleAnnotation` should either be `c("test","ctrl")` or `c("ctrl","test")`.
`geneIdCol`	Column name (or number of column) in `rowData` of `x`, i.e. `SummarizedExperiment` object, that represents the gene ID of the introns and exons in `x`.
`silent`	Whether run the function silently, i.e. without printing the top differential expression tags.
`group`	Vector to manually define the sample groups (or annotations). It is ignored if `sampleAnnopCol` is defined.
`rejection.region`	The `rejection.region` parameter in `exactTest` from `edgeR` package.
`big.count`	The `big.count` parameter in `exactTest` from `edgeR` package.
`prior.count`	The `prior.count` parameter in `exactTest` from `edgeR` package.
`disp`	The type of estimating the dispersion in the data. Available options are: "tagwise", "trended", "common" and "genewise". It is also possible to assign a number for manually setting the `disp`.
`...`	Other parameter settings for the `estimateDisp` function (e.g. the `design` parameter) in the `edgeR` package.

Value

`table`	Data frame containing columns for the log2 fold-change (logFC), the average of log2 counts-per-million (logCPM), and the two-sided p-value (PValue).
`comparison`	The name of the two compared groups.
`dispersionType`	The name of the type of dispersion used.
`dispersion`	The estimated dispersion values.

Author(s)

Ali Oghabian

See Also

`lfc`, `glmInterest`, `qlfInterest`, `treatInterest`, `DEXSeqIntEREst`

Examples

```
geneId<- paste("gene", c(rep(1,5), rep(2,5), rep(3,5), rep(4,5)),
sep="_")
readCnt1<- sample(1:100, 20)
readCnt2<- sample(1:100, 20)
readCnt3<- sample(1:100, 20)
readCnt4<- sample(1:100, 20)
fpkm1<- readCnt1/(tapply(readCnt1, geneId, sum))[geneId]
fpkm2<- readCnt2/(tapply(readCnt2, geneId, sum))[geneId]
fpkm3<- readCnt3/(tapply(readCnt3, geneId, sum))[geneId]
fpkm4<- readCnt4/(tapply(readCnt4, geneId, sum))[geneId]

# Creating object using test data
interestDat<- data.frame(
int_ex=rep(c(rep(c("exon","intron"),2),"exon"),4),
int_ex_num= rep(c(1,1,2,2,3),4),
gene_id= geneId,
sam1_readCnt=readCnt1,
sam2_readCnt=readCnt2,
sam3_readCnt=readCnt3,
sam4_readCnt=readCnt4,
sam1_fpkm=fpkm1,
sam2_fpkm=fpkm2,
sam3_fpkm=fpkm3,
sam4_fpkm=fpkm4
)
readFreqColIndex<- grep("_readCnt$",colnames(interestDat))
scaledRetentionColIndex<- grep("_fpkm$",colnames(interestDat))

scalRetTmp<- as.matrix(interestDat[ ,scaledRetentionColIndex])
colnames(scalRetTmp)<-gsub("_fpkm$","", colnames(scalRetTmp))

frqTmp<- as.matrix(interestDat[ ,readFreqColIndex])
colnames(frqTmp)<-gsub("_readCnt$","", colnames(frqTmp))


InterestResultObj<- InterestResult(
resultFiles=paste("file",1:4, sep="_"),
rowData= interestDat[ , -c(readFreqColIndex,
scaledRetentionColIndex)],
counts= frqTmp,
scaledRetention= scalRetTmp,
scaleLength=TRUE,
scaleFragment=FALSE,
sampleAnnotation=data.frame(
sampleName=paste("sam",1:4, sep=""),
gender=c("M","M","F","F"), row.names=paste("sam", 1:4, sep="")
)
)

res<- exactTestInterest(InterestResultObj, sampleAnnoCol="gender",
sampleAnnotation=c("F","M"), geneIdCol= "gene_id",
silent=TRUE, disp="common")
```

`getRepeatTable` *Get table of regions with repetetive DNA sequences*

Description

This function returns a data.frame that includes regions with repetetive DNA sequences. These sequences can bias the mapping of the reads to the genome excluding them will remove the bias.

Usage

```
getRepeatTable( dbUser="genome",
dbHost="genome-mysql.cse.ucsc.edu",ucscGenome="hg19",
ucscTable="rmsk", minLength=0, repFamilyFil="Alu",
repFamilyCol="repFamily", repChrCol="genoName",
repBegCol="genoStart", repEndCol="genoEnd",
repStrandCol="strand", repNameCol="repName",
repClassCol="repClass")
```

Arguments

`dbUser`	Database user name; set as "genome" by default.
`dbHost`	Database host address; set as "genome-mysql.cse.ucsc.edu" by default.
`ucscGenome`	The UCSC genome.
`ucscTable`	The UCSC table name. The table with repetetive sequences by default it is set as `"rmsk"`.
`minLength`	the minimum length criteria to consider the repetetive sequences. the default setting is `0`.
`repFamilyFil`	A vector including the repeats family to consider. By default the `"Alu"` elements are considered.
`repFamilyCol`	The name of the column of the input table (`ucscTable`) that represents the repeats family.
`repChrCol`	The column (either name or the number of the column) of the input table that represents the Chromosome names.
`repBegCol`	The column of the table that represents the start coordinates.
`repEndCol`	The column of the table that represents the end coordinates.
`repStrandCol`	The column of the table that represents the strand.
`repNameCol`	The column of the table representing the repeats' names.
`repClassCol`	The column of the table representing the repeats' classes.

Value

Data frame with columns representing coordinates and annotations of repetitive DNA elements.

Author(s)

Ali Oghabian

Examples

```
## Not run:
# Download table for Alu elemnts in the human genome
suppressWarnings(repTable<- getRepeatTable(repFamilyFil="Alu",
ucscGenome="hg19"))

## End(Not run)
```

`glmInterest` *generalized linear model likelihood ratio tests*

Description

Compute generalized linear model likelihood ratio tests using `edgeR` package. For more information see `glmfit` and `glmLRT()` functions in `edgeR` package.

Usage

```
glmInterest(x, design=c(), silent=TRUE, disp="common",
coef=c(), contrast=NULL, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`design`	Design matrix.
`silent`	Whether run the function silently, i.e. without printing the top differential expression tags. Default is TRUE.
`disp`	The method of estimating the dispersion in the data. Available options are: "common", "trended", "tagwiseInitCommon" and "tagwiseInitTrended". It is also possible to assign a number.
`coef`	Integer or character vector indicating which coefficients of the linear model are to be tested equal to zero. See `glmLRT()` in `edgeR` for more information.
`contrast`	Numeric vector or matrix specifying contrasts of the linear model coefficients to be tested equal to zero. See `glmLRT()` in `edgeR` for more information.
`...`	Other parameter settings for the `glmLRT()` function in the `edgeR` package.

Value

All values produced by `glmLRT` in `edgeR` package plus following:

`dispersionType`	The name of the type of dispersion used.
`dispersion`	The estimated dispersion values.

Author(s)

Ali Oghabian

See Also

`exactTestInterest`, `qlfInterest`, `treatInterest`

Examples

```
#Test retention differentiation across the 3 types of sampels
group <- getAnnotation(mdsChr22Obj)[,"type"]
glmRes<- glmInterest(x=mdsChr22Obj,
design=model.matrix(~group), silent=TRUE,
disp="tagwiseInitTrended", coef=2:3, contrast=NULL)
```

`interest` *Wrapper function: Parallel run*

Description

A read summarization function that countsns all the reads mapping to the introns/exons based on the users detailed parameter settings. The process can be run in parallel on multiple computing cores to improve it performance.

Usage

```
interest( bamFileYieldSize=1000000, bamFile, isPaired,
isPairedDuplicate=FALSE, isSingleReadDuplicate= NA, reference,
referenceGeneNames, referenceIntronExon, repeatsTableToFilter=c(),
junctionReadsOnly=FALSE, outFile, logFile="",
returnObj= FALSE, method=c("IntRet", "ExEx", "IntSpan"),
clusterNo=NULL, bpparam, appendLogFile=FALSE, sampleName="",
scaleLength= c(TRUE,FALSE), scaleFragment= c(TRUE,TRUE), ...)
```

Arguments

`bamFileYieldSize`
: Maximum number of pair reads in the temprorary files created as the result of dividing the input .bam file.

`bamFile`
: Path of the input bam file.

`isPaired`
: Whether the bam file is the result of a paired end sequencing read mapping (TRUE) or not (FALSE).

`isPairedDuplicate`
: Whether extract only (if set TRUE), filter (FALSE) or include (if set NA) PCR dupplicates for paired mapped reads. It uses the FLAG field in the bam file to filter the duplicate read. If the mapping software does not support detection and flaging the duplicate reads `dedup` tool of `BamUtil` or `MarkDuplicates` of `Picard` tools could be used.

`isSingleReadDuplicate`
: Whether extract only (if set TRUE), filter (FALSE) or include (if set NA) PCR dupplicates for single mapped reads.

`reference`
: Dataframe to be used as reference; It should at least contain three same-size vectors with the tag names `chr`, `begin`, and `end` which describe the exons and introns genome coordinates. It also accepts a `GRanges` object. To build a new reference check the `referencePrepare` function.

`referenceGeneNames`
: A vector with the same size as the row-size of the reference which includes the gene names of the reference.

`referenceIntronExon`
: A vector with the same size as the row-size of the reference with values "intron" and "exon" describing which (intron or exon) each row of the reference represents.

`repeatsTableToFilter`
: A data.frame table with similar stucture to the `reference`. It includes `chr`, `begin`, and `end` columns. If defined, all reads mapped to the described regions would be ingnored and the Intron/exon lengths would be corrected to exclude the to exclude the regions with repetitive DNA sequences. See `getRepeatTable`.

`junctionReadsOnly`
: The parameter is considered if the `method` is set as `IntRet` or `ExEx` (NOT `IntSpan`). It declares whether only consider the Intron-Exon or Exon-Exon junction reads and ignore the reads that fully map to exons or introns. By default this argument is set as FALSE.

`outFile`
: The name or path of the result file.

`logFile`
: The log file path; if defined log information are written to the log file.

`returnObj`
: If set `TRUE` in addition to making result text files, the results would also be returned as an object of class `SummarizedExperiment`.

`method`
: A vector describing the summarization methods to use; i.e. whether count reads mapping to the introns (`IntRet`), reads mapping to the exons (`ExEx`), or reads spanning the introns (`IntSpan`). In `IntSpan` mode the introns in the reference are taken into account only; while in `IntRet` the introns and their spanning exons, and in `ExEx` mode only the exons in the reference are taken into account.

`clusterNo`	Number of parallel cluster nodes. As default (clusterNo=NULL) the total number of CPUs that are available in the cluster would be used.
`bpparam`	An optional `BiocParallelParam` instance defining the parallel back-end to be used.
`appendLogFile`	Whether log information should be appended to the `logFile`. It is set `FALSE` by default.
`sampleName`	The name of the sample being analyzed. It will be included in the returned object if `returnObj` is `TRUE`.
`scaleLength`	A vector constructed of TRUE/FALSE values, same size as the `method` argument. It indcates whether the retention levels of the intron/exons should be scaled to their lengths.
`scaleFragment`	A vector constructed of TRUE/FALSE values, same size as the `method` argument. It indcates whether the retention levels of the intron/exons should be scaled to the sum of retention levels (i.e. mapped fragments) over the genes.
`...`	Other parameter settings specific to `BamFile-class` function in the `Rsamtools` package. Parameters `qnamePrefixEnd` and `qnameSuffixStart` are in particular useful to modify qnames in the BAM files.

Value

If `returnObj` is set `TRUE` in addition to making result text files, dependant on whether a single or two `method` is defined, the results would be returned as a single object of class `SummarizedExperiment` or as a list of size 2 which includes 2 objects of class `SummarizedExperiment` one for IntRet and the other for ExEx.

Author(s)

Ali Oghabian

See Also

`interest.sequential`.

Examples

```
# Creating temp directory to store the results
outDir<- file.path(tempdir(),"interestFolder")
dir.create(outDir)
outDir<- normalizePath(outDir)

# Loading suitable bam file
bamF <- system.file("extdata", "small_test_SRR1691637_ZRSR2Mut_RHBDD3.bam",
package="IntEREst", mustWork=TRUE)

# Choosing reference for the gene RHBDD3
ref= u12[u12[,"gene_name"]=="RHBDD3",]

test= interest(
bamFileYieldSize=10000,
bamFile=bamF,
isPaired=TRUE,
isPairedDuplicate=FALSE,
isSingleReadDuplicate=NA,
reference=ref,
referenceGeneNames=ref[,"ens_gene_id"],
referenceIntronExon=ref[,"int_ex"],
repeatsTableToFilter=c(),
outFile=paste(outDir,
    "interestRes.tsv", sep="/"),
logFile=paste(outDir,
    "log.txt", sep="/"),
method=c("IntRet", "IntSpan"),
junctionReadsOnly=FALSE,
clusterNo=1,
returnObj=TRUE,
scaleLength= c(TRUE,FALSE),
scaleFragment= c(TRUE,TRUE)
)

test
```

`interest.sequential` *Wrapup function: Sequential running*

Description

A read summarization function that countsns all the reads mapping to the introns/exons based on the users detailed parameter settings. The process runs on a single computing core.

Usage

```
interest.sequential( bamFileYieldSize=1000000, bamFile, isPaired,
isPairedDuplicate=FALSE, isSingleReadDuplicate=NA,
reference, referenceGeneNames,
referenceIntronExon, repeatsTableToFilter=c(),
junctionReadsOnly=FALSE, outFile, logFile="",
returnObj= FALSE, method=c("IntRet", "ExEx", "IntSpan"),
appendLogFile=FALSE, sampleName="",
scaleLength= c(TRUE,FALSE), scaleFragment= c(TRUE,TRUE), ...)
```

Arguments

`bamFileYieldSize`	Maximum number of paired Reads in the temprorary files created as the result of dividing the input .bam file.
`bamFile`	Path of the input bam file.

`isPaired`	Whether the bam file is the result of a paired end sequencing read mapping (TRUE) or not (FALSE).
`isPairedDuplicate`	Whether extract only (if set TRUE), filter (FALSE) or include (if set NA) PCR dupplicates for paired mapped reads. It uses the FLAG field in the bam file to filter the duplicate read. If the mapping software does not support detection and flaging the duplicate reads `dedup` tool of `BamUtil` or `MarkDuplicates` of `Picard` tools could be used.
`isSingleReadDuplicate`	Whether extract only (if set TRUE), filter (FALSE) or include (if set NA) PCR dupplicates for single mapped reads.
`reference`	Dataframe to be used as reference; It should at least contain three same-size vectors with the tag names `chr`, `begin`, and `end` which describe the genome coordinates of the introns and exons. It also accepts a `GRanges` object as input. To build a new reference check the `referencePrepare` function.
`referenceGeneNames`	A vector with the same size as the row-size of the reference which include the gene names.
`referenceIntronExon`	A vector with the same size as the row-size of the reference with values "intron" and "exon" describing which (intron or exon) each row of the reference represents.
`repeatsTableToFilter`	A data frame with similar structure as the `reference`, i.e. includes `chr`, `begin`, and `end` columns. If defined, all reads mapped to the described regions would be ingnored and the Intron/exon lengths would be corrected to exclude the regions with repetetive DNA sequences. See `getRepeatTable`.
`junctionReadsOnly`	The parameter is considered if the `method` is set as `IntRet` or `ExEx` (NOT `IntSpan`). It declares whether only consider the Intron-Exon or Exon-Exon junction reads and ignore the reads that fully map to exons or introns. By default this argument is set as FALSE.
`outFile`	The name or path of the result file.
`logFile`	The log file path; if defined log information are written to the log file.
`returnObj`	If set `TRUE` in addition to producing result text files, the results would also be returned as an object of class `SummarizedExperiment`.
`method`	A vector describing the summarization methods to use; i.e. whether count reads mapping to the introns (`IntRet`), reads mapping to the exons (`ExEx`), or reads spanning the introns (`IntSpan`). In `IntSpan` mode the introns in the reference are taken into account only; while in `IntRet` the introns and their spanning exons, and in `ExEx` mode only the exons in the reference are taken into account.
`appendLogFile`	Whether log information should be appended to the `logFile`. It is `FALSE` by default.
`sampleName`	The name of the sample being analyzed. It will be included in the returned object if `returnObj` is `TRUE`.

`scaleLength`	A vector constructed of TRUE/FALSE values, same size as the `method` argument. It indcates whether the retention levels of the intron/exons should be scaled to their lengths.
`scaleFragment`	A vector constructed of TRUE/FALSE values, same size as the `method` argument. It indcates whether the retention levels of the intron/exons should be scaled to the sum of retention levels (i.e. mapped fragments) over the genes.
`...`	Other parameter settings specific to `BamFile-class` function in the `Rsamtools` package. Parameters `qnamePrefixEnd` and `qnameSuffixStart` are in particular useful to modify qnames in the BAM files.

Value

If `returnObj` is set `TRUE` in addition to making result text files, dependant on whether a single or two `method` is defined, the results would be returned as a single object of class `SummarizedExperiment` or as a list of size 2 which includes 2 objects of class `SummarizedExperiment` one for IntRet and the other for ExEx.

Author(s)

Ali Oghabian

See Also

`interest`.

Examples

```
# Creating temp directory to store the results
outDir<- file.path(tempdir(),"interestFolder")
dir.create(outDir)
outDir<- normalizePath(outDir)

# Loading suitable bam file
bamF <- system.file("extdata", "small_test_SRR1691637_ZRSR2Mut_RHBDD3.bam",
package="IntEREst", mustWork=TRUE)


# Choosing reference for the gene RHBDD3
ref=u12[u12[,"gene_name"]=="RHBDD3",]

test= interest.sequential(
bamFileYieldSize=10000,
bamFile=bamF,
isPaired=TRUE,
isPairedDuplicate=FALSE,
isSingleReadDuplicate=NA,
reference=ref,
referenceGeneNames=ref[,"ens_gene_id"],
referenceIntronExon=ref[,"int_ex"],
repeatsTableToFilter=c(),
outFile=paste(outDir,
    "interestRes.tsv", sep="/"),
logFile=paste(outDir,
    "log.txt", sep="/"),
method=c("IntRet","IntSpan"),
returnObj=TRUE,
scaleLength= c(TRUE,FALSE),
scaleFragment= c(TRUE,TRUE)
)

test
```

`InterestResult` *Building SummarizedExperiment object from results in IntEREst.*

Description

Calls the constructors and creates a `SummarizedExperiment` object. For more information on the resulted object and the class see `SummarizedExperiment-class`.

Usage

```
InterestResult(resultFiles=c(), counts, scaledRetention,
scaleLength, scaleFragment, sampleAnnotation, rowData)
```

Arguments

`resultFiles`	Vector of link to the result files of `interest`.
`counts`	Numeric Matrix that includes the read counts.
`scaledRetention`	Matrix that includes the scaled retention values.
`scaleLength`	Logical value, indicating whether the intron/exon retention levels are scaled to the length of the introns/exons.
`scaleFragment`	Logical value, indicating whether the intron/exon retention levels are scaled to the fragments mapped to the genes.
`sampleAnnotation`	Data frame with the row-size equal to the size of `resultFiles` and `sampleAnnotation`. Each column of the matrix represents annotations for the samples. Column name represents annotation name.
`rowData`	Data frame with Intron/Exon annotations and read count and scaled retention values for each sample.

Value

Returns an object of class `SummarizedExperiment`.

Author(s)

Ali Oghabian

See Also

SummarizedExperiment-class attributes addAnnotation counts-method plot-method

Examples

```
geneId<- paste("gene", c(rep(1,5), rep(2,5), rep(3,5), rep(4,5)),
sep="_")
readCnt1<- sample(1:100, 20)
readCnt2<- sample(1:100, 20)
readCnt3<- sample(1:100, 20)
readCnt4<- sample(1:100, 20)
fpkm1<- readCnt1/(tapply(readCnt1, geneId, sum))[geneId]
fpkm2<- readCnt2/(tapply(readCnt2, geneId, sum))[geneId]
fpkm3<- readCnt3/(tapply(readCnt3, geneId, sum))[geneId]
fpkm4<- readCnt4/(tapply(readCnt4, geneId, sum))[geneId]

# Creating object using test data
interestDat<- data.frame(
int_ex=rep(c(rep(c("exon","intron"),2),"exon"),4),
int_ex_num= rep(c(1,1,2,2,3),4),
gene_id= geneId,
sam1_readCnt=readCnt1,
sam2_readCnt=readCnt2,
sam3_readCnt=readCnt3,
sam4_readCnt=readCnt4,
sam1_fpkm=fpkm1,
sam2_fpkm=fpkm2,
sam3_fpkm=fpkm3,
sam4_fpkm=fpkm4
)
readFreqColIndex<- grep("_readCnt$",colnames(interestDat))
scaledRetentionColIndex<- grep("_fpkm$",colnames(interestDat))

scalRetTmp<- as.matrix(interestDat[ ,scaledRetentionColIndex])
colnames(scalRetTmp)<-gsub("_fpkm$","", colnames(scalRetTmp))

frqTmp<- as.matrix(interestDat[ ,readFreqColIndex])
colnames(frqTmp)<-gsub("_readCnt$","", colnames(frqTmp))


InterestResultObj<- InterestResult(
resultFiles=paste("file",1:4, sep="_"),
rowData= interestDat[ , -c(readFreqColIndex,
scaledRetentionColIndex)],
counts= frqTmp,
scaledRetention= scalRetTmp,
scaleLength=TRUE,
scaleFragment=FALSE,
sampleAnnotation=data.frame(
sampleName=paste("sam",1:4, sep=""),
gender=c("M","M","F","F"), row.names=paste("sam", 1:4, sep="")
)
)

# View object
InterestResultObj
```

---

interestResultIntEx *Building results object that contains Intron-retention and exon-exon junction information*

Description

Building SummarizedExperiment-class object from an intron retention and an exon-exon junction results in IntEREst. The average of the junction levels are added to the SummerizedExperiment object of the intron retentions.

Usage

```
interestResultIntEx (intObj, exObj, intExCol=c(),
mean.na.rm=TRUE, postExName="ex_junc" )
```

Arguments

`intObj`	A `SummarizedExperiment` including intron retention information.
`exObj`	A `SummarizedExperiment` including exon-exon junction information.
`intExCol`	Column name (or number) in the `rowData` of the intron object that represents whether each row of `x` assays is "intron" or "exon".
`mean.na.rm`	Whether exclude missing values when measuring the mean.
`postExName`	The postfix to use for the column names of the exons junction values in the

Value

Returns an object of class `SummarizedExperiment`.

Author(s)

Ali Oghabian

See Also

`SummarizedExperiment-class` `attributes` `addAnnotation` `counts-method` `plot-method`

Examples

```
testIntObj<- InterestResult(
resultFiles= paste(paste("testFile",1:3, sep="_"),"bam", sep="."),
counts= matrix(1:15, ncol=3, nrow=5, byrow=TRUE,
dimnames= list(c(), paste("s", 1:3, sep="_"))),
scaledRetention= matrix(1:15, ncol=3, nrow=5, byrow=TRUE,
dimnames= list(c(), paste("s", 1:3, sep="_"))),
scaleLength= FALSE,
scaleFragment= FALSE,
sampleAnnotation= data.frame(
files=paste(paste("testFile",1:3, sep="_"),"bam", sep="."),
names=paste("s", 1:3, sep="_"),
row.names=paste("s", 1:3, sep="_")),
rowData=data.frame(id= paste("i", 1:5, sep="_"),
chr= rep("chr1", 5),
begin=seq(100, by=100, length.out=5 ),
end=seq(110, by=100, length.out=5 ),
strand=rep("+",5))
)

testExObj<- InterestResult(
resultFiles= paste(paste("testFile",1:3, sep="_"),"bam", sep="."),
counts= matrix(1:30, ncol=3, nrow=10, byrow=TRUE,
dimnames= list(c(), paste("s", 1:3, sep="_"))),
scaledRetention= matrix(1:30, ncol=3, nrow=10, byrow=TRUE,
dimnames= list(c(), paste("s", 1:3, sep="_"))),
scaleLength= FALSE,
scaleFragment= FALSE,
sampleAnnotation= data.frame(
files=paste(paste("testFile",1:3, sep="_"),"bam", sep="."),
names=paste("s", 1:3, sep="_"),
row.names=paste("s", 1:3, sep="_")),
rowData=data.frame(id= paste("e", 1:10, sep="_"),
chr= rep("chr1", 10),
begin= c(seq(90, by=100, length.out=5),
seq(111, by=100, length.out=5)),
end= c(seq(99, by=100, length.out=5),
seq(120, by=100, length.out=5 )),
strand=rep("+",10))
)


(testIntExObj<- interestResultIntEx(intObj=testIntObj, exObj=testExObj,
mean.na.rm=TRUE, postExName="ex_junc" ) )
```

`intexIndex` *Extract index of intron or exon rows*

Description

Extract row numbers where introns (or exons dependant on user's request) are located in an object of type `SummarizedExperiment`.

Usage

```
intexIndex(x, intExCol="int_ex", what="intron")
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`intExCol`	Column name (or number) that represents whether each row is "intron" or "exon" in `rowData` of `x`.
`what`	A character string that defines whether the index for the introns or exons should be returned. Accepts either `"exon"` or `"intron"` (default) as values.

Value

A numeric vector which includes the index of the introns/exons.

Author(s)

Ali Oghabian

See Also

`u12NbIndex`

Examples

```
# Show the few first index of rows that represent the introns
head(intexIndex(mdsChr22Obj, what="intron"))
```

`lfc` *Log fold change*

Description

Log fold change estimation and normalized log fold change using `edgeR` package.

Usage

```
lfc(x, fcType="edgeR", sampleAnnoCol=c(), sampleAnnotation=c(),
silent=TRUE, group=c(), rejection.region="doubletail",
pseudoCnt=1, log2=TRUE, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`fcType`	Available as "scaledRetention" or "edgeR" (as default) corresponding to either log fold change of scaled retention values or degeR normalized log fold change values.
`sampleAnnoCol`	Which colummn of `colData` of `x` to consider for the analysis.
`sampleAnnotation`	A vector of size 2 which cotains values from `colData` of `SummarizedExperiment` object; e.g. if `getAnnotation(x)[,sampleAnnoCol]= c("test","test","ctrl","ctrl",...)` , and the goal is to compare "test" and "ctrl" samples, `sampleAnnotation` should either be `c("test","ctrl")` or `c("ctrl","test")`.
`silent`	Whether run `exactTestInterest` silently, without warnings.
`group`	Vector to manually define the sample groups (or annotations). It is ignored if `sampleAnnoCol` is defined.
`rejection.region`	The rejection.region parameter in `exactTest`, considered only if `fcType` is "edgeR".
`pseudoCnt`	Pseudo count for log transformation (default=1).
`log2`	Logical value either TRUE (default) or FALSE indicating whether the fold-changes should be log 2 transformed.
`...`	Other parameter settings from the `exactTestInterest` function.

Value

Vector including fold change values.

Author(s)

Ali Oghabian

See Also

`exactTestInterest`, `u12DensityPlotIntron`

Examples

```
lfcFpkm<- lfc(mdsChr22Obj, fcType="scaledRetention",
sampleAnnoCol="test_ctrl",
sampleAnnotation=c("ctrl", "test"),
silent=TRUE, group=c(), pseudoFpkm=1, log2=TRUE)


lfcEdgeRFpkm<- lfc(mdsChr22Obj, fcType="edgeR",
sampleAnnoCol="test_ctrl",
sampleAnnotation=c("ctrl", "test"),
silent=TRUE, group=c(), pseudoFpkm=1, log2=TRUE)
```

`mdsChr22ExObj` *Object of SummarizedExperiment type for exon-exon junction of MDS data*

Description

The Results of `interest()` analysis in exon-exon junction mode, for the genes that feature U12-type introns and are located on Chr22 in MDS data.

Usage

```
data(mdsChr22ExObj)
```

Format

An Object of class `SummarizedExperiment` that contains intron retention results generated by `interest()` function on MDS data consisting of bone-marrows samples of 8 MDS patients with ZRSR2 mutations, 4 patients without the mutation and 4 healthy individuals.

`@colData` A "DataFrame" (from "S4Vectors" package) that its rownames can be set as the sample identification names and the other columns are various annotations for the samples. Its column names are characters that describe the annotations.

`@assays` List of size 2 that includes two numeric matrices: `counts` that includes raw read counts of the sequencing reads mapped to introns and exons, and (2) `scaledRetention`, i.e. the normalized read counts.

`@NAMES` A NULL value.

`@elementMetadata` A "DataFrame" (from "S4Vectors" package) that include intron and exon annotations.

`@metadata` A list of size 2 that includes parameter settings for the `interest()` and `interest.sequential()` runs.

Value

Object of class `SummarizedExperiment`.

Source

Madan, V., et.al., Aberrant splicing of U12-type introns is the hallmark of ZRSR2 mutant myelodysplastic syndrome. Nat Communication 2015 Jan 14;6:6042. doi: 10.1038/ncomms7042.

`mdsChr22IntSpObj` *Object of SummarizedExperiment type for intron spanning reads of MDS data*

Description

The Results of `interest()` analysis in intron-spanning mode, for the genes that feature U12-type introns and are located on Chr22 in MDS data.

Usage

```
data(mdsChr22ExObj)
```

Format

An Object of class `SummarizedExperiment` that contains intron retention results generated by `interest()` function on MDS data consisting of bone-marrows samples of 8 MDS patients with ZRSR2 mutations, 4 patients without the mutation and 4 healthy individuals.

`@colData` A "DataFrame" (from "S4Vectors" package) that its rownames can be set as the sample identification names and the other columns are various annotations for the samples. Its column names are characters that describe the annotations.

`@assays` List of size 2 that includes two numeric matrices: `counts` that includes raw read counts of the sequencing reads mapped to introns and exons, and (2) `scaledRetention`, i.e. the normalized read counts.

`@NAMES` A NULL value.

`@elementMetadata` A "DataFrame" (from "S4Vectors" package) that include intron and exon annotations.

`@metadata` A list of size 2 that includes parameter settings for the `interest()` and `interest.sequential()` runs.

Value

Object of class `SummarizedExperiment`.

Source

Madan, V., et.al., Aberrant splicing of U12-type introns is the hallmark of ZRSR2 mutant myelodysplastic syndrome. Nat Communication 2015 Jan 14;6:6042. doi: 10.1038/ncomms7042.

`mdsChr22Obj` *Object of SummarizedExperiment type for intron retention MDS data*

Description

The Results of `interest()` analysis in Intron-retention mode, for the genes that feature U12-type introns and are located on Chr22 in MDS data.

Usage

```
data(mdsChr22Obj)
```

Format

An Object of class `SummarizedExperiment` that contains intron retention results generated by `interest()` function on MDS data consisting of bone-marrows samples of 8 MDS patients with ZRSR2 mutations, 4 patients without the mutation and 4 healthy individuals.

`@colData` A "DataFrame" (from "S4Vectors" package) that its rownames can be set as the sample identification names and the other columns are various annotations for the samples. Its column names are characters that describe the annotations.

`@assays` List of size 2 that includes two numeric matrices: `counts` that includes raw read counts of the sequencing reads mapped to introns and exons, and (2) `scaledRetention`, i.e. the normalized read counts.

`@NAMES` A NULL value.

`@elementMetadata` A "DataFrame" (from "S4Vectors" package) that include intron and exon annotations.

`@metadata` A list of size 2 that includes parameter settings for the `interest()` and `interest.sequential()` runs.

Value

Object of class `SummarizedExperiment`.

Source

Madan, V., et.al., Aberrant splicing of U12-type introns is the hallmark of ZRSR2 mutant myelodysplastic syndrome. Nat Communication 2015 Jan 14;6:6042. doi: 10.1038/ncomms7042.

mergeInterestResult *merge two SummarizedExperiment objects into one*

Description

Build a new object bu merging data of two `SummarizedExperiment` objects.

Usage

```
mergeInterestResult(x, y)
```

Arguments

x	Object of type `SummarizedExperiment`.
y	Object of type `SummarizedExperiment`.

Value

An object of calss `SummarizedExperiment`.

Author(s)

Ali Oghabian

See Also

`interest`, `InterestResult`.

Examples

```

geneId<- paste("gene", c(rep(1,5), rep(2,5), rep(3,5), rep(4,5)),
sep="_")
readCnt1<- sample(1:100, 20)
readCnt2<- sample(1:100, 20)
readCnt3<- sample(1:100, 20)
readCnt4<- sample(1:100, 20)
fpkm1<- readCnt1/(tapply(readCnt1, geneId, sum))[geneId]
fpkm2<- readCnt2/(tapply(readCnt2, geneId, sum))[geneId]
fpkm3<- readCnt3/(tapply(readCnt3, geneId, sum))[geneId]
fpkm4<- readCnt4/(tapply(readCnt4, geneId, sum))[geneId]

# Creating object using test data
interestDat<- data.frame(
int_ex=rep(c(rep(c("exon","intron"),2),"exon"),4),
int_ex_num= rep(c(1,1,2,2,3),4),
gene_id= geneId,
sam1_readCnt=readCnt1,
sam2_readCnt=readCnt2,
sam3_readCnt=readCnt3,
sam4_readCnt=readCnt4,
sam1_fpkm=fpkm1,
sam2_fpkm=fpkm2,
sam3_fpkm=fpkm3,
sam4_fpkm=fpkm4
)
readFreqColIndex<- grep("_readCnt$",colnames(interestDat))
scaledRetentionColIndex<- grep("_fpkm$",colnames(interestDat))

scalRetTmp<- as.matrix(interestDat[ ,scaledRetentionColIndex])
colnames(scalRetTmp)<-gsub("_fpkm$","", colnames(scalRetTmp))

frqTmp<- as.matrix(interestDat[ ,readFreqColIndex])
colnames(frqTmp)<-gsub("_readCnt$","", colnames(frqTmp))

#Object including data for Males
interestResObjM<-InterestResult(
resultFiles=paste("file",1:2, sep="_"),
rowData= interestDat[, -c(readFreqColIndex,
scaledRetentionColIndex)],
counts= frqTmp[,1:2],
scaledRetention= scalRetTmp[,1:2],
scaleLength=TRUE,
scaleFragment=FALSE,
sampleAnnotation=data.frame(
sampleName=paste("sam",1:2, sep=""),
gender=c("M","M"),
health=c("healthy","unhealthy"),
row.names=paste("sam", 1:2, sep="")
)
)

#Object including data for Females
interestResObjF<-InterestResult(
resultFiles=paste("file",3:4, sep="_"),
rowData= interestDat[, -c(readFreqColIndex,
scaledRetentionColIndex)],
counts= frqTmp[,3:4],
scaledRetention= scalRetTmp[,3:4],
scaleLength=TRUE,
scaleFragment=FALSE,
sampleAnnotation=data.frame(
sampleName=paste("sam",3:4, sep=""),
gender=c("F","F"),
health=c("healthy","unhealthy"),
row.names=paste("sam", 3:4, sep="")
)
)

#Build new object
newObj<- mergeInterestResult(interestResObjM, interestResObjF)
```

```
#View newObj
print(newObj)
```

plot-method *plot - method*

Description

plot method for `SummarizedExperiment` objects.

Usage

```
## S4 method for signature 'SummarizedExperiment,ANY'
plot(x, summary="none",
subsetRows=NULL, what="scaled", intronExon="intron",
logScaleBase=NULL, logPseudoCnt=1, plotLoess=TRUE,
loessCol="red", loessLwd=1, loessLty=1, cexText=1,
marPlot=c(2,2,2,2), mgpPlot=c(1, 1, 0), cexAxis=1,
writeCor=TRUE, corCex=1, corMethod="pearson", corCol="grey63",
upperCorXY=c("topleft", NULL), lowerCorXY=c("topleft", NULL),
na.rm=TRUE,cex=1, sampleAnnoCol=c(), lowerPlot=FALSE,
upperPlot=TRUE, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment` generated by either `interest()`, `interest.sequential()` or `readInterestResults()`.
`summary`	Whether to plot the mean or median of the values over the sample with the same annotations, or plot the values for each individual sample separately. The available options are "mean", "median", or "none".
`subsetRows`	Vector either constructed of TRUE/FALSE values or constructed of numeric values that could be used to choose rows of `x` i.e. the `SummarizedExperiment` object.
`what`	Whether plot "scaled" (default) or read counts ("counts").
`intronExon`	Whether plot intron retention, i.e. "intron" (default) or exon-junction "exon".
`logScaleBase`	Base of the log transform of the values, if defined. By default the value is `NULL` meaning that the values would not be log transformed.
`logPseudoCnt`	Pseudocount for the log transformation (default=1).
`plotLoess`	Whether fit and plot LOESS curve line (default="red").
`loessCol`	loess line colour (default="red").
`loessLwd`	loess line width (default=1).
`loessLty`	loess line type (default=1).
`cexText`	Size of the text for sample names or annotations (default=1).

`marPlot`	Plot margins (default=c(2,2,2,2)). See `?par` for more information.
`mgpPlot`	Plotting `mgp` parameter (default=c(1, 1, 0)). See `?par` for more information.
`cexAxis`	Size of the text for the axis (default=1).
`writeCor`	Write correlation values (default=TRUE).
`corCex`	Text size of correlation values (default=1).
`corMethod`	Method used for correlation calculation. For more information see `cor` from `stats` package of R.
`corCol`	Color of the text of correlation (default="grey").
`upperCorXY`	The coordinates of the correlation text in the upper panel plots (default= c("topleft", NULL)).
`lowerCorXY`	The coordinates of the correlation text in the lower panel plots (default= c("topleft", NULL)).
`na.rm`	whether remove the rows with missing values (default=TRUE).
`cex`	size of the plot text and symbols (default=1).
`sampleAnnoCol`	Which colummn of `colData` of object `SummarizedExperiment` to consider for plotting.
`lowerPlot`	Whether plot the lower panel (default=FALSE).
`upperPlot`	Whether plot the upper panel (default=TRUE).
`...`	Other arguments to pass to the `plot()` function.

Value

Returns NULL.

Author(s)

Ali Oghabian

See Also

Class: `SummarizedExperiment-class` Method: `counts-method` `boxplot-method`

Examples

```
geneId<- paste("gene", c(rep(1,5), rep(2,5), rep(3,5), rep(4,5)),
sep="_")
readCnt1<- sample(1:100, 20)
readCnt2<- sample(1:100, 20)
readCnt3<- sample(1:100, 20)
readCnt4<- sample(1:100, 20)
fpkm1<- readCnt1/(tapply(readCnt1, geneId, sum))[geneId]
fpkm2<- readCnt2/(tapply(readCnt2, geneId, sum))[geneId]
fpkm3<- readCnt3/(tapply(readCnt3, geneId, sum))[geneId]
fpkm4<- readCnt4/(tapply(readCnt4, geneId, sum))[geneId]
```

```
# Creating object using test data
interestDat<- data.frame(
int_ex=rep(c(rep(c("exon","intron"),2),"exon"),4),
int_ex_num= rep(c(1,1,2,2,3),4),
gene_id= geneId,
sam1_readCnt=readCnt1,
sam2_readCnt=readCnt2,
sam3_readCnt=readCnt3,
sam4_readCnt=readCnt4,
sam1_fpkm=fpkm1,
sam2_fpkm=fpkm2,
sam3_fpkm=fpkm3,
sam4_fpkm=fpkm4
)
readFreqColIndex<- grep("_readCnt$",colnames(interestDat))
scaledRetentionColIndex<- grep("_fpkm$",colnames(interestDat))

scalRetTmp<- as.matrix(interestDat[ ,scaledRetentionColIndex])
colnames(scalRetTmp)<-gsub("_fpkm$","", colnames(scalRetTmp))

frqTmp<- as.matrix(interestDat[ ,readFreqColIndex])
colnames(frqTmp)<-gsub("_readCnt$","", colnames(frqTmp))


InterestResultObj<- InterestResult(
resultFiles=paste("file",1:4, sep="_"),
rowData= interestDat[ , -c(readFreqColIndex,
scaledRetentionColIndex)],
counts= frqTmp,
scaledRetention= scalRetTmp,
scaleLength=TRUE,
scaleFragment=FALSE,
sampleAnnotation=data.frame(
sampleName=paste("sam",1:4, sep=""),
gender=c("M","M","F","F"), row.names=paste("sam", 1:4, sep="")
)
)

InterestResultObj2<- addAnnotation(x=InterestResultObj,
sampleAnnotationType="health",
sampleAnnotation=c("healthy","unhealthy","healthy","unhealthy")
)

#Plotting
plot(InterestResultObj)
plot(InterestResultObj, sampleAnnoCol="gender", summary="mean")
plot(InterestResultObj2, sampleAnnoCol=3, summary="mean")
plot(InterestResultObj2, summary="none")
```

psi *Psi values estimation*

Description

Calculating the relative inclusion level of intron or Psi values base on two count matrices from a single or two separate objects. The values for each intron is in the range of [0,1], where 0 means complete splicing or no retention of the intron and 1 represnet complete 100

Usage

```
psi (x, y, intCol, exCol, pseudoCnt=0)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`y`	Optional; i.e. an object of type `SummarizedExperiment`.
`intCol`	Column numbers or column names in counts matrix of `x` which include the number of reads mapped to the introns.
`exCol`	Column numbers or column names in counts matrix of `x` (or if defined `y`) which include the number of reads spanning the introns (or mapping exons flanking the introns).
`pseudoCnt`	Pseudo counts to sum to the denominator of the devision to avoid devision to zero.

Value

data.frame with column size equal to the size of `intCol` parameter, and row size equal to the number of rows in `x`. It contains the psi values (i,e.values between 0 and 1 showing the fraction of spliced in transcripts).

Author(s)

Ali Oghabian

See Also

`interestResultIntEx`

Examples

```
mdsChr22IntObj<- mdsChr22Obj[which(rowData(mdsChr22Obj)$int_ex=="intron"), ]

#Build object including intron-retention and exon-junction results
mdsChr22RefIntExObj<- interestResultIntEx(intObj=mdsChr22Obj,
exObj=mdsChr22ExObj, mean.na.rm=TRUE, postExName="ex_junc",
intExCol="int_ex" )
# Calculate Psi
psiRes<- psi(mdsChr22RefIntExObj,
intCol=which(colData(mdsChr22RefIntExObj)$intronExon=="intron"),
exCol=which(colData(mdsChr22RefIntExObj)$intronExon=="exon"))
# show Psi results
head(psiRes)
```

`pwmU12db` *PWM of U12 and U2-type introns splice sites*

Description

PWM of U12 and U2-type introns splice sites and it is based on the U12DB database.

Usage

```
data("pwmU12db")
```

Format

A list that contains Position Weight Matrices (PWM) of donor site, branch point and acceptor site of U12-type introns and the PWMs of donor site and acceptor site of U2-type introns. It is based on the U12DB database.

`pwmDonU12` A position weigh matrix for the donor site of the U12-type introns, with 4 rows and 46 columns. The rows of the matrix represent "A", "C", "G", and "T" nucleotides and the columns represent the postions in the genome. Each position in the matrix include a weight (i.e. number between 0 and 1) which indicates how common the corresponding base (represented by the row of the matrix) is observed in the corerespoding position (represented by the colum of the matrix).

`pwmBpU12` A position weigh matrix for the branch point of the U12-type introns, with 4 rows and 9 columns.

`pwmAccU12` A position weigh matrix for the acceptor site of the U12-type introns, with 4 rows and 46 columns.

`pwmDonU2` A position weigh matrix for the donor site of the U2-type introns, with 4 rows and 25 columns.

`pwmAccU2` A position weigh matrix for the acceptor site of the U12-type introns, with 4 rows and 46 columns.

Value

List of 5 numeric matrices representing the PWMs of donor site of U12-type introns, branch point site of U12-type introns, acceptor site of U12-type introns, donor site of U2-type introns, and acceptor site of U2-type introns.

Source

Alioto, T.S. U12DB: a database of orthologous U12-type spliceosomal introns. Nucleic Acids Research 2006, doi: 10.1093/nar/gkl796

qlfInterest *quasi-likelihood F-test*

Description

Compute quasi-likelihood F-test using edgeR package. For more information see glmQLFit and glmQLFTest functions in edgeR package.

Usage

```
qlfInterest(x, design=c(), silent=TRUE, disp="common",
coef=c(), contrast=NULL,
poisson.bound=TRUE, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`design`	Design matrix.
`silent`	Whether run silently, i.e. without printing the top differential expression tags. The default is TRUE.
`disp`	The method of estimating the dispersion in the data. Available options are: "common", "trended", "tagwiseInitCommon" and "tagwiseInitTrended". It is also possible to assign a number.
`coef`	Integer or character vector indicating which coefficients of the linear model are to be tested equal to zero. See `glmQLFTest` for more information.
`contrast`	Numeric vector or matrix specifying contrasts of the linear model coefficients to be tested equal to zero. See `glmQLFTest` for more information.
`poisson.bound`	Logical value, if TRUE (i.e. default) the pvalue would be higher than when obtained fom likelihood ratio test while Negative Binomial dispersion is zero.
`...`	Other parameter settings for the `glmQLFTest` function in the `edgeR` package.

Value

All values produced by `glmQLFTest` plus the following :

`dispersionType`	The name of the type of dispersion used.
`dispersion`	The estimated dispersion values.

Author(s)

Ali Oghabian

See Also

`exactTestInterest`, `glmInterest`, `treatInterest`

Examples

```
#Test retention differentiation across the 3 types of sampels
group <- getAnnotation(mdsChr22Obj)[,"type"]
qlfRes<- qlfInterest(x=mdsChr22Obj,
design=model.matrix(~group), silent=TRUE,
disp="tagwiseInitTrended", coef=2:3, contrast=NULL)

qlfRes
```

`readInterestResults` *Read interest/interest.sequential results text files*

Description

Reads one or multiple text file results generated by the `interest` or `interest.sequential` functions and builds an object of `SummarizedExperiment-class` class.

Usage

```
readInterestResults(resultFiles, sampleNames,
sampleAnnotation, commonColumns, freqCol, scaledRetentionCol,
scaleLength, scaleFragment, reScale=FALSE, geneIdCol,
repeatsTableToFilter=c())
```

Arguments

`resultFiles`	Vector of character strings which includes the path to the tab-separated files resulted by the `interest` function.
`sampleNames`	Vector of character strings which includes the name of the samples. It should be the same size as the `resultFiles` parameter.
`sampleAnnotation`	Data frame with the same row number as the size of `resultFiles` and `sampleNames` parameter. The column names represent the annotation names and values in each column represent the annotations of the samples.
`commonColumns`	Columns in the result file which include intron/exon annotations and are common across all files defined in `resultFiles`.
`freqCol`	Column in the result file which include the read counts for introns/exons.
`scaledRetentionCol`	Column in the result file which include the scaled retention values for introns/exons.
`scaleLength`	Logical value, indicating whether the intron/exon retention levels are scaled to the length of the introns/exons. If `reScale` is `TRUE` the scaled retention levels would be rescalculated when reading the data.
`scaleFragment`	Logical value, indicating whether the intron/exon retention levels are scaled to the fragments mapped to the genes. If `reScale` is `TRUE` the scaled retention levels would be rescalculated when reading the data.

reScale	Logical value, indicating whether the scaled retention levels would be rescalculated when reading the data. By default it does not calculate and trusts the user to set the scaleLength and scaleFragment parameters correctly, i.e. as it was set in the interest() or interest.sequential() analysis.
geneIdCol	The number or name of the column in resultFiles which represents the gene/transcript names. It would be used for summing up the number of mapped fragments to the genes when scaling the retention levels. It is only used if reScale and scaleFragment arguments are set TRUE.
repeatsTableToFilter	A data.frame table with similar stucture to the reference. It includes chr, begin, and end columns. If defined, all reads mapped to the described regions would be ingnored and the Intron/exon lengths would be corrected to exclude the to exclude the regions with repetitive DNA sequences. See getRepeatTable. It is only used if reScale and scaleLength arguments are set TRUE.

Value

An object of calss SummarizedExperiment-class.

Author(s)

Ali Oghabian

See Also

interest, InterestResult.

Examples

```
geneId<- paste("gene", c(rep(1,7), rep(2,7), rep(3,7), rep(4,7)),
sep="_")
readCnt1<- sample(1:100, 28)
readCnt2<- sample(1:100, 28)
readCnt3<- sample(1:100, 28)
readCnt4<- sample(1:100, 28)
fpkm1<- readCnt1/(tapply(readCnt1, geneId, sum))[geneId]
fpkm2<- readCnt2/(tapply(readCnt2, geneId, sum))[geneId]
fpkm3<- readCnt3/(tapply(readCnt3, geneId, sum))[geneId]
fpkm4<- readCnt4/(tapply(readCnt4, geneId, sum))[geneId]

#Create tmp director
tmpDir=file.path(tempdir(),"InterestResult")
dir.create(tmpDir)

# Build text files similar to files resulted by interest
dfTmp=data.frame(
int_ex=rep(c(rep(c("exon","intron"),3),"exon"),4),
int_ex_num= rep(c(1,1,2,2,3,3,4),4),
int_type=rep(c(NA,"U2",NA,"U12",NA,"U2",NA),4),
strand=rep("*",28),
gene_id= geneId,
sam1_readCnt=readCnt1,
sam2_readCnt=readCnt2,
sam3_readCnt=readCnt3,
sam4_readCnt=readCnt4,
sam1_fpkm=fpkm1,
sam2_fpkm=fpkm2,
sam3_fpkm=fpkm3,
sam4_fpkm=fpkm4
)

writeDf<-function(df, file){
write.table(df, file, col.names=TRUE,
row.names=FALSE, quote=FALSE, sep='\t')
}

writeDf(dfTmp[, c(1:5,6,10)], paste(tmpDir, "df1.tsv", sep="/"))
writeDf(dfTmp[, c(1:5,7,11)], paste(tmpDir, "df2.tsv", sep="/"))
writeDf(dfTmp[, c(1:5,8,12)], paste(tmpDir, "df3.tsv", sep="/"))
writeDf(dfTmp[, c(1:5,9,13)], paste(tmpDir, "df4.tsv", sep="/"))

# Build object from generated text file results
testObj<-readInterestResults(
resultFiles=paste(tmpDir,
c("df1.tsv", "df2.tsv", "df3.tsv", "df4.tsv"), sep="/"),
sampleNames=c("sam1","sam2","sam3","sam4"),
sampleAnnotation= data.frame( gender=c("M","M","F","F"),
health=c("healthy","unhealthy","healthy","unhealthy")),
commonColumns=1:5, freqCol=6, scaledRetentionCol=7,
scaleLength=FALSE, scaleFragment=TRUE, reScale=FALSE)

#View object
testObj
```

referencePrepare *Creates reference file*

Description

Creates reference file for IntEREst functions, e.g. `interest()`. The function uses functions of `biomaRt` library.

Usage

```
referencePrepare( outFileTranscriptsAnnotation="",
annotateGeneIds=TRUE,
u12IntronsChr=c(), u12IntronsBeg=c(), u12IntronsEnd=c(),
u12IntronsRef,collapseExons=TRUE, sourceBuild="UCSC",
ucscGenome="hg19", ucscTableName="knownGene",
ucscUrl="http://genome-euro.ucsc.edu/cgi-bin/",
biomart="ENSEMBL_MART_ENSEMBL",
biomartDataset="hsapiens_gene_ensembl",
biomartTranscriptIds=NULL, biomartExtraFilters=NULL,
biomartIdPrefix="ensembl_",biomartHost="www.ensembl.org",
biomartPort=80,circSeqs="", miRBaseBuild=NA, taxonomyId=NA,
filePath="", fileFormat=c("auto", "gff3", "gtf"), fileDatSrc=NA,
fileOrganism=NA, fileChrInf=NULL,
fileDbXrefTag=c(), addCollapsedTranscripts=TRUE,
ignore.strand=FALSE )
```

Arguments

`outFileTranscriptsAnnotation`	If defined outputs transcripts annotations.
`annotateGeneIds`	Wether annotate and add the gene ids information.
`collapseExons`	Whether collapse (i.e. reduce) the exonic regions. TRUE by default.
`sourceBuild`	The source to use to build the reference data, `"UCSC"`, `"biomaRt"`, and `"file"` (for GFF3 or GTF files) are supported.
`ucscGenome`	The genome to use. `"hg19"` is the default. See `genome` parameter of `makeTxDbFromUCSC` function of `GenomicFeatures` library for more information.
`ucscTableName`	The UCSC table name to use. See `tablename` parameter of `makeTxDbFromUCSC` function of `GenomicFeatures` library for more information.
`ucscUrl`	The UCSC URL address. See `url` parameter of `makeTxDbFromUCSC` function of `GenomicFeatures` library for more information.
`u12IntronsChr`	A vector of character strings that includes chromsomal locations of the U12 type introns. If defined together with `u12IntronsBeg` and `u12IntronsBeg`, they would be used to annotate the U12-type introns.
`u12IntronsBeg`	A vector of numbers that defines the begin (or start) coordinates of the u12-type introns.
`u12IntronsEnd`	A vector of numbers that defines the end coordinates of the u12-type introns.
`u12IntronsRef`	A GRanges object that includes the coordinates of the U12 type introns. If defined, it would be used to annotate the U12-type introns.
`biomart`	BioMart database name. See `biomart` parameter of `makeTxDbFromBiomart` function of `GenomicFeatures` library for more information.
`biomartDataset`	BioMart dataset name; default is "hsapiens_gene_ensembl". See `dataset` parameter of `makeTxDbFromBiomart` function of `GenomicFeatures` library for more information.
`biomartTranscriptIds`	optional parameter to only retrieve transcript annotation results for a defined set of transcript ids. See `transcript_ids` parameter of `makeTxDbFromBiomart` function of `GenomicFeatures` library for more information.

`biomartExtraFilters`	A list of names; i.e. additional filters to use in the BioMart query. See `filters` parameter of `makeTxDbFromBiomart` function of `GenomicFeatures` library for more information.
`biomartIdPrefix`	A list of names; i.e. additional filters to use in the BioMart query. See `id_prefix` parameter of `makeTxDbFromBiomart` function of `GenomicFeatures` library for more information.
`biomartHost`	Host to connect to; the default is "www.ensembl.org". For older versions of the GRCH you can provide the archive websites, e.g. for GRCH37 you can use "grch37.ensembl.org".
`biomartPort`	The port to use in the HTTP communication with the host. Default is 80.
`circSeqs`	A character vector that includes chromosomes that should be marked as circular. See `circ_seqs` parameter of `makeTxDbFromBiomart` and `makeTxDbFromUCSC` functions of `GenomicFeatures` library for more information.
`miRBaseBuild`	Set appropriate build Information from mirbase.db to use for microRNAs (default=NA). See `miRBaseBuild` parameter of `makeTxDbFromBiomart` and `makeTxDbFromUCSC` functions of `GenomicFeatures` library for more information.
`taxonomyId`	This parameter can be used to provide taxonomy Ids. It is set to NA by default. You can check the taxonomy Ids with the `available.species()` function in `GenomeInfoDb` package. For more information see `taxonomyId` parameter of `makeTxDbFromBiomart` and `makeTxDbFromUCSC` functions of `GenomicFeatures` library.
`filePath`	Character string i.e. the path to file. Used if `sourceBuild` is `"file"`.
`fileFormat`	The format of the input file. `"auto"`, `"gff3"` and `"gtf"` is supported.
`fileDatSrc`	Character string describing the source of the data file. Used if `sourceBuild` is `"file"`.
`fileOrganism`	The genus and species name of the organism. Used if `sourceBuild` is `"file"`.
`fileChrInf`	Dataframe that includes information about the chromosome. The first column represents the chromosome name and the second column is the length of the chromosome. Used if `sourceBuild` is `"file"`.
`fileDbXrefTag`	A vector of chracater strings which if defined it would be used as feature names. Used if `sourceBuild` is `"file"`.
`addCollapsedTranscripts`	Whether add a column that includes the collapsed transcripts information. Used if `collapseExons` is `TRUE`.
`ignore.strand`	Whether consider the strands in the reference. If set `TURE` the strands would be ingnored.

Value

Data frame that includes the coordinates and annotations of the introns and exons of the transcripts, i.e. the reference.

Author(s)

Ali Oghabian

Examples

```
# Build test gff3 data
tmpGen<- u12[u12[,"ens_trans_id"]=="ENST00000413811",]
tmpEx<-tmpGen[tmpGen[,"int_ex"]=="exon",]
exonDat<- cbind(tmpEx[,3], ".",
tmpEx[,c(7,4,5)], ".", tmpEx[,6], ".",paste("ID=exon",
tmpEx[,11], "; Parent=ENST00000413811", sep="") )
trDat<- c(tmpEx[1,3], ".", "mRNA", as.numeric(min(tmpEx[,4])),
as.numeric(max(tmpEx[,5])), ".", tmpEx[1,6], ".",
"ID=ENST00000413811")

outDir<- file.path(tempdir(),"tmpFolder")
dir.create(outDir)
outDir<- normalizePath(outDir)

gff3File=paste(outDir, "gffFile.gff", sep="/")

cat("##gff-version 3\n",file=gff3File, append=FALSE)
cat(paste(paste(trDat, collapse="\t"),"\n", sep=""),
file=gff3File, append=TRUE)

write.table(exonDat, gff3File,
row.names=FALSE, col.names=FALSE,
sep='\t', quote=FALSE, append=TRUE)

# Selecting U12 introns info from 'u12' data
u12Int<-u12[u12$int_ex=="intron"&u12$int_type=="U12",]

# Test the function
refseqRef<- referencePrepare (sourceBuild="file",
filePath=gff3File, u12IntronsChr=u12Int[,"chr"],
u12IntronsBeg=u12Int[,"begin"],
u12IntronsEnd=u12Int[,"end"], collapseExons=TRUE,
fileFormat="gff3", annotateGeneIds=FALSE)
```

`subInterestResult` *Extract subset of object*

Description

Build a new object using subset of data in an `SummarizedExperiment` object.

Usage

```
subInterestResult(x, selectRow, selectCol,
sampleAnnoCol, sampleAnnotation=c())
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`selectRow`	Numeric or TRUE/FALSE Vector indicating what rows to extract.
`selectCol`	A vector with Numeric values, character strings (sample names) or TRUE/FALSE Vector indicating what columns to extract.
`sampleAnnoCol`	Which colummn of `colData` of object `x` to consider for subset data extraction.
`sampleAnnotation`	Vector including the annotations to consider for subset data extraction. They should be present in the `sampleAnnoCol` column of the `colData` of `x`.

Value

An object of calss `SummarizedExperiment`.

Author(s)

Ali Oghabian

See Also

`interest`, `InterestResult`.

Examples

```
geneId<- paste("gene", c(rep(1,7), rep(2,7), rep(3,7), rep(4,7)),
sep="_")
readCnt1<- sample(1:100, 28)
readCnt2<- sample(1:100, 28)
readCnt3<- sample(1:100, 28)
readCnt4<- sample(1:100, 28)
fpkm1<- readCnt1/(tapply(readCnt1, geneId, sum))[geneId]
fpkm2<- readCnt2/(tapply(readCnt2, geneId, sum))[geneId]
fpkm3<- readCnt3/(tapply(readCnt3, geneId, sum))[geneId]
fpkm4<- readCnt4/(tapply(readCnt4, geneId, sum))[geneId]

# Creating object using test data
interestDat<-data.frame(
int_ex=rep(c(rep(c("exon","intron"),3),"exon"),4),
int_ex_num= rep(c(1,1,2,2,3,3,4),4),
int_type=rep(c(NA,"U2",NA,"U12",NA,"U2",NA),4),
strand=rep("*",28),
gene_id= geneId,
sam1_readCnt=readCnt1,
sam2_readCnt=readCnt2,
sam3_readCnt=readCnt3,
sam4_readCnt=readCnt4,
sam1_fpkm=fpkm1,
sam2_fpkm=fpkm2,
sam3_fpkm=fpkm3,
sam4_fpkm=fpkm4
)
readFreqColIndex<- grep("_readCnt$",colnames(interestDat))
scaledRetentionColIndex<- grep("_fpkm$",colnames(interestDat))
samNames<-paste("sam", 1:4, sep="")
frqTmp<-as.matrix(interestDat[, readFreqColIndex])
sclTmp<-as.matrix(interestDat[, scaledRetentionColIndex])
colnames(frqTmp)<- samNames
colnames(sclTmp)<- samNames
interestResObj<- InterestResult(
resultFiles=paste("file",1:4, sep="_"),
rowData= interestDat[, -c(readFreqColIndex,
scaledRetentionColIndex)],
counts= frqTmp,
scaledRetention= sclTmp ,
scaleLength=TRUE,
scaleFragment=FALSE,
sampleAnnotation=data.frame(
sampleName=paste("sam",1:4, sep=""),
gender=c("M","M","F","F"),
health=c("healthy","unhealthy","healthy","unhealthy"),
row.names=samNames
)
)

#Build new object
newObj<- subInterestResult(interestResObj, selectRow=1:20)

#View newObj
print(newObj)
```

`treatInterest` *Differential retention test relative to a threshold*

Description

Compute a genewise statistical test relative to a fold-change threshold using edgeR package. For more information see `glmTreat` function in edgeR package.

Usage

```
treatInterest(x, design=c(), silent=TRUE, disp="common",
coef=c(), contrast=NULL, lfc=0, ...)
```

Arguments

`x`	Object of class `SummarizedExperiment`.
`design`	Design matrix.

`silent`	Whether run silently, i.e. without printing the top differential expression tags. Default is TRUE.
`disp`	The method of estimating the dispersion in the data. Available options are: "common", "trended", "tagwiseInitCommon" and "tagwiseInitTrended". It is also possible to assign a number.
`coef`	Integer or character vector indicating which coefficients of the linear model are to be tested equal to zero. See `glmTreat` for more information.
`contrast`	Numeric vector or matrix specifying contrasts of the linear model coefficients to be tested equal to zero. See `glmTreat` for more information.
`lfc`	Numeric scalar i.e. the log fold change threshold.
`...`	Other parameter settings for the `glmFit` function in the `edgeR` package.

Value

All values produced by `glmTreat` plus the following :

`dispersionType`	The name of the type of dispersion used.
`dispersion`	The estimated dispersion values.

Author(s)

Ali Oghabian

See Also

`exactTestInterest`, `qlfInterest`, `glmInterest`

Examples

```
group <- getAnnotation(mdsChr22Obj)[,"type"]

#Test retention differentiation across the 3 types of sampels
# The log fold change threshold is 0
treatRes<- treatInterest(x=mdsChr22Obj,
design=model.matrix(~group), silent=TRUE,
disp="tagwiseInitTrended", coef=2:3, contrast=NULL, lfc=0)
treatRes
```

---

u12 *U12 data*

Description

Intron/exon annotations of genes featuring U12 introns. It is based on HG19/GRCh37 (converted from hg17/NCBI35). Moreover the u12 genes are based on the U12DB database.

Usage

```
data("u12")
```

Format

A data frame with 22713 observations on the following 17 variables.

`id`	a numeric vector
`int_ex_id`	a character vector
`chr`	a character vector
`begin`	a numeric vector
`end`	a numeric vector
`strand`	a numeric vector
`int_ex`	a character vector
`trans_type`	a character vector
`ens_gene_id`	a character vector
`ens_trans_id`	a character vector
`int_ex_num`	a numeric vector
`gene_name`	a character vector
`trans_name`	a character vector
`overlap_no`	a numeric vector
`int_type`	a character vector
`int_subtype`	a character vector

Value

Data frame that includes the coordinates and annotations of the introns and exons of the transcripts, i.e. the reference.

Source

Alioto, T.S. U12DB: a database of orthologous U12-type spliceosomal introns. Nucleic Acids Research 2006, doi: 10.1093/nar/gkl796

u12Boxplot *U12 boxplot*

Description

A boxplot method for U12 and U2-type introns of `SummarizedExperiment` objects.

Usage

```
u12Boxplot(x, sampleAnnoCol=NA, intExCol="int_ex",
intTypeCol="int_type", intronExon, col="white",
boxplotNames=c(), lasNames=3, outline=FALSE, addGrid=FALSE, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`sampleAnnoCol`	Which colummn of `colData` in `x` to consider for plotting.
`intExCol`	Column name (or number) that represents whether each row of `x` assays is "intron" or "exon".
`intTypeCol`	Column name (or number) that represents what type of intron each row of `x` assays represents.
`intronExon`	Whether plot intron retention (set `intronExon="intron"`) or exon-exon junction (set `intronExon="exon"`) levels.
`col`	Vector showing box colours. It is either of size 1 or the same size as the number of groups to be plotted.
`boxplotNames`	Names to write under boxes. If not defined, as names, it pastes U12/U2 (intron annotation) to the sample group annotations separated by a space " ".
`lasNames`	Orientation of the box names.
`outline`	If outline is TRUE the outlier points are drawn otherwise if FALSE (default) they are not.
`addGrid`	Whether add a grid under the boxplots (FALSE by default).
`...`	Other arguments to pass to the `boxplot()` function.

Value

A `SummarizedExperiment` object.

Author(s)

Ali Oghabian

See Also

`u12BoxplotNb`

Examples

```
u12Boxplot(mdsChr22Obj, sampleAnnoCol="type",
    intExCol="int_ex",  intTypeCol="intron_type", intronExon="intron",
    col=rep(c("orange", "yellow"),3) ,  lasNames=3,
    outline=FALSE, ylab="FPKM", cex.axis=0.8)
```

---

u12BoxplotNb   *boxplot U12 introns retention levels (or flanking exons junction levels) and (up/down)stream U2 introns (or exons junction levels)*

Description

boxplot U12 introns and (Up/Down)stream U2 introns in `SummarizedExperiment` objects.

Usage

```
u12BoxplotNb(x, sampleAnnoCol=2, intExCol="int_ex",
intTypeCol="int_type", intronExon, strandCol="strand", geneIdCol,
col=c(), names=c(), lasNames=1, outline=FALSE, plotLegend=TRUE,
cexLegend=1, xLegend="topright", yLegend=NULL, bgLegend="transparent",
legend=c(), addGrid=FALSE, ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`sampleAnnoCol`	Which colummn of `colData` of `x` to consider for plotting.
`intExCol`	Column name (or number) that represents whether each row of `x` assays is "intron" or "exon".
`intTypeCol`	Column name (or number) that represents what type of intron each row of `x` assays represents.
`intronExon`	Whether plot intron retention (set `intronExon="intron"`) or exon-exon junction (set `intronExon="exon"`) levels.
`strandCol`	Column name (or number) that represents the strand of each row of assays in `x`. The values in the column are either "+", "-" or "*".
`geneIdCol`	Column name (or number) that represents the gene ID of each row of assays in `x`.
`col`	Vector containing box colours. It is either of size 1 or the same size as the number of boxes resulted based on the grouping of the samples defined by `sampleAnnoCol`.
`names`	Names to write under group of boxes.
`lasNames`	Orientation of the box names.

`outline`	If outline is TRUE the outlier points are drawn otherwise if FALSE (default) they are not.
`plotLegend`	Whether show legend (TRUE by default).
`cexLegend`	Size of the text in legend .
`xLegend, yLegend`	Position of legend in the plot. For more info see `x` and `y` parameters in `legend`.
`bgLegend`	Bakcground colour of the legend box. It is "transparent" by default.
`legend`	The replacement texts to be used in legend.
`addGrid`	Whether add a grid under the boxplots (FALSE by default).
`...`	Other arguments to pass to the `boxplot()` function.

Value

Returns NULL

Author(s)

Ali Oghabian

See Also

`u12Boxplot`

Examples

```
u12BoxplotNb(mdsChr22Obj, sampleAnnoCol="type", lasNames=1,
    intExCol="int_ex", intTypeCol="intron_type", intronExon="intron",
    boxplotNames=c(), outline=FALSE, plotLegend=TRUE,
    geneIdCol="collapsed_transcripts_id", xLegend="topleft",
    col=c("pink", "lightblue", "lightyellow"), ylim=c(0,600000),
    ylab="FPKM", cex.axis=0.8)
```

`u12DensityPlot` *Density plot of fld changes of intron retention and exon-exon junction levels*

Description

Density plot of fold change of the retention levels of U12- vs U2- type intron, or exon-exon junction levels of the flanking exons. For the density plot of the foldchange of intron retention levels the `u12DensityPlotIntron()` function or `u12DensityPlot()` function with `intronExon= "intron"` can be used. For density plot of the foldchange of exon-exon junction levels use `u12DensityPlot()` function with `intronExon= "exon"`.

Usage

```
u12DensityPlot(x,
type=c("U12", "U2Up", "U2Dn", "U2UpDn", "U2Rand"),
fcType="edgeR", sampleAnnotation=c(),  sampleAnnoCol=c(),
group=c(), intExCol="int_ex", intTypeCol="int_type", intronExon,
strandCol="strand", geneIdCol="collapsed_transcripts",
naUnstrand=FALSE, col=1, lty=1, lwd=1, plotLegend=TRUE,
cexLegend=1, xLegend="topright", yLegend=NULL, legend=c(),
randomSeed=NULL, xlab="", ...)

u12DensityPlotIntron(x,
type= c("U12", "U2Up", "U2Dn", "U2UpDn", "U2Rand"),
fcType= "edgeR", sampleAnnotation=c(),  sampleAnnoCol=c(),
group=c(), intExCol="int_ex", intTypeCol="int_type",
strandCol= "strand", geneIdCol= "collapsed_transcripts",
naUnstrand=FALSE, col=1, lty=1, lwd=1, plotLegend=TRUE,
cexLegend=1, xLegend="topright", yLegend=NULL, legend=c(),
randomSeed=NULL, xlab="", ...)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`type`	A vector that includes the type of introns to plot. Available options are U12 introns "U12", U2 introns at downstream of U12 introns "U2Dn", U2 introns at upstream of U12 introns "U2Up", U2 introns at upstream or downstream of U12 introns suitable for when the coorduinates in object x are unstranded (their strand is "*") "U2UpDn", random U2 introns from object x "U2Rand". Settings "U2Up", "U2Dn" and "U2UpDn" are useful only if the refernce is linearly ordered. References with exons only resulted by `referencePrepare` and `unionRefTr` are NOT necessarily linearly ordered.
`fcType`	Available as "fpkm" or "edgeR" (as default) corresponding to either log fold change of fpkm values or degeR normalized log fold change values.
`sampleAnnoCol`	Which colummn of `colData` of `x` to consider for plotting.
`sampleAnnotation`	A vector of size 2 which cotains values from `colData` of `SummarizedExperiment` object; e.g. if `getAnnotation(x)[,sampleAnnoCol]= c("test","test","ctrl","ctrl",...)` , and the goal is to compare "test" and "ctrl" samples, `sampleAnnotation` should either be `c("test","ctrl")` or `c("ctrl","test")`.
`group`	Vector to manually define the sample groups (or annotations). It is ignored if `sampleAnnoCol` is defined.
`intExCol`	Column name (or number) that represents whether each row of `x` assays is "intron" or "exon".
`intTypeCol`	Column name (or number) that represents what type of intron each row of `x` assays represents.
`intronExon`	Whether plot intron retention (set `intronExon="intron"`) or exon-exon junction (set `intronExon="exon"`) levels.

`strandCol`	Column name (or number) that represents the strand of each row of assays in `x`. The values in the column are either "+", "-" or "*".
`geneIdCol`	Column name (or number) that represents the gene ID of each row of assays in `x`.
`naUnstrand`	Replace unstranded results, i.e. introns or exon with "*" strand, with NA (to be excluded).
`col`	A vector with the size of 1 or the same size as the `type` parameter which includes the colour/colours of the plotted density lines (default=1).
`lty`	A vector with the size of 1 or the same size as the `type` parameter which includes the type of the plotted density lines (default=1).
`lwd`	A vector with the size of 1 or the same size as the `type` parameter which includes the width of the plotted density lines (default=1).
`plotLegend`	Whether show legend (TRUE by default).
`cexLegend`	Size of the text in legend .
`xLegend, yLegend`	Position of legend in the plot. For more info see `x` and `y` parameters in `legend`.
`legend`	The replacement texts to be used in legend.
`randomSeed`	Seed value for random number generator.
`xlab`	The lable of the X axis of the plot; by default it is "".
`...`	Other parameter settings from the `plot` function.

Value

Returns NULL.

Author(s)

Ali Oghabian

See Also

`exactTestInterest`, `lfc`

Examples

```
u12DensityPlotIntron(mdsChr22Obj,
    type= c("U12", "U2Up", "U2Dn", "U2UpDn", "U2Rand"),
    fcType= "edgeR", sampleAnnoCol="test_ctrl",
    sampleAnnotation=c("ctrl","test"), intExCol="int_ex",
    intTypeCol="intron_type", strandCol= "strand",
    geneIdCol= "collapsed_transcripts_id", naUnstrand=FALSE, col=c(2,3,4,5,6),
    lty=c(1,2,3,4,5), lwd=1, plotLegend=TRUE, cexLegend=0.7,
    xLegend="topright", yLegend=NULL, legend=c(), randomSeed=10,
    ylim=c(0,0.6), xlab=expression("log"[2]*" fold change FPKM"))
```

u12Index *Extract index of U12 introns rows*

Description

Extract row numbers of U12 introns in an object of class `SummarizedExperiment`.

Usage

```
u12Index(x, intExCol="int_ex", intTypeCol="int_type", intronExon="intron")
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`intExCol`	Column name (or number) that represents whether each row of `x` assays is "intron" or "exon".
`intTypeCol`	Column name (or number) that represents what type of intron each row of `x` assays represents.
`intronExon`	Whether extract U12 type introns (set `intronExon="intron"`) or exon-exon junction (set `intronExon="exon"`) flanking U12 introns.

Value

A numeric vector which includes the index of U12 introns.

Author(s)

Ali Oghabian

See Also

`u12NbIndex`

Examples

```
head(u12Index(mdsChr22Obj, intTypeCol="intron_type"))
```

u12NbIndex *Extract index of U2 introns (up/down)stream of U12 introns rows*

Description

Extract row numbers of U2-type introns (up/down)stream of U12-type introns (in the `@interestDf` attribute of an object of class `SummarizedExperiment`).

Usage

```
u12NbIndex(x, intExCol="int_ex", intTypeCol="int_type",
strandCol="strand", geneIdCol="collapsed_transcripts",
naUnstrand=FALSE)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`intExCol`	Column name (or number) that represents whether each row of `x` assays is "intron" or "exon".
`intTypeCol`	Column name (or number) that represents what type of intron each row of `x` assays represents.
`strandCol`	Column name (or number) that represents the strand of each row of assays in `x`. The values in the column are either "+", "-" or "*".
`geneIdCol`	Column name (or number) that represents the gene ID of each row of assays in `x`.
`naUnstrand`	Replace unstranded results, i.e. introns or exon with "*" strand, with NA. If set as FALSE (default) "*" strand would be same as "+" strand.

Value

`upIntron`	A numeric vector which includes the index of U2-type intron upstream the U12-type introns.
`downIntron`	A numeric vector which includes the index of U2-type intron downstream the U12-type introns.
`upExon`	A numeric vector which includes the index of exon upstream the U12-type introns.
`downExon`	A numeric vector which includes the index of exon downstream the U12-type introns.

Author(s)

Ali Oghabian

See Also

`u12Index`

Examples

```
head(u12NbIndex(mdsChr22Obj, intExCol="int_ex",
intTypeCol="intron_type", strandCol="strand",
geneIdCol="collapsed_transcripts_id", naUnstrand=FALSE))
# Return NA if no strand information available
head(u12NbIndex(mdsChr22Obj, intExCol="int_ex",
intTypeCol="intron_type", strandCol="strand",
geneIdCol="collapsed_transcripts_id", naUnstrand=TRUE))
```

`unionRefTr` *Union introns/exons of transcripts*

Description

Performs union on the overlapping introns/exons so that the final merged transcripts would feature from each exon or intron, one copy.

Usage

```
unionRefTr( referenceChr, referenceBegin, referenceEnd, referenceTr,
referenceIntronExon, intronExon="exon", silent=FALSE)
```

Arguments

`referenceChr`	Chromosome names of the references (e.g. introns).
`referenceBegin`	A vector that corresponds to the begin coordinates of the reference.
`referenceEnd`	A vector that corresponds to the end coordinates of the reference.
`referenceTr`	A character vector that includes transcription IDs.
`referenceIntronExon`	A vector with the same size as the `referenceChr`, `referenceBegin` and `referenceEnd` which contains `'intron'` and `'exon'` describing what (either intron or exon) each element of the 3 vectors represents.
`intronExon`	Should be assigned either `'intron'` or `'exon'` or `c('intron','exon')` based on whether match the PWM to the intronic, exonic, or intronic and exonic regions of the reference. By default it seeks matches in intronic regions (intronExon='intron').
`silent`	Whether run silently.

Value

Data frame containing merged transcripts structure. The merged transcripts feature from each intron or exon, one copy ONLY.

Author(s)

Ali Oghabian

See Also

annotateU12.

Examples

```
unU12Ex<-unionRefTr( referenceChr=u12[1:94,"chr"],
referenceBegin=u12[1:94,"begin"], referenceEnd=u12[1:94,"end"],
referenceTr=u12[1:94,"trans_name"],
referenceIntronExon=u12[1:94,"int_ex"], intronExon="exon", silent=TRUE)

unU12Int<-unionRefTr( referenceChr=u12[1:94,"chr"],
referenceBegin=u12[1:94,"begin"], referenceEnd=u12[1:94,"end"],
referenceTr=u12[1:94,"trans_name"],
referenceIntronExon=u12[1:94,"int_ex"], intronExon="intron", silent=TRUE)

unU12IntEx<-unionRefTr( referenceChr=u12[1:94,"chr"],
referenceBegin=u12[1:94,"begin"], referenceEnd=u12[1:94,"end"],
referenceTr=u12[1:94,"trans_name"],
referenceIntronExon=u12[1:94,"int_ex"], intronExon=c("intron","exon"),
silent=TRUE)
```

`updateRowDataCol` *Updating contents of* `rowData` *of* `SummarizedExperiment` *objects*

Description

Updates the values in a single column of the `rowData` of `SummarizedExperiment` objects.

Usage

```
updateRowDataCol(x,  updateCol, value)
```

Arguments

`x`	Object of type `SummarizedExperiment`.
`updateCol`	Name or the number of the column in the `rowData` of `x` to be updated with the new values. if the `updateCol` does not match to any column names it will be added as a new column.
`value`	The new Replacing values.

Value

Returns an object of type `SummarizedExperiment`.

Author(s)

Ali Oghabian

See Also

annotateU12

Examples

```
test<- mdsChr22Obj
# See the the frequency of each intron type annotation
table(rowData(test)$intron_type)

#Change U2 to u2
newIntType<- as.character(rowData(test)$intron_type)
newIntType[newIntType=="U2" &
!is.na(newIntType=="U2")]<- "u2"
#Updating values
test<- updateRowDataCol(test, updateCol="intron_type",
value=newIntType)
#See the frequency of the updated intron type annotations
table(rowData(test)$intron_type)

#Adding a new column
test<- updateRowDataCol(test, updateCol="new_column",
value=rep(NA, nrow(rowData(test))) )
head(rowData(test))
```

Index