

Package ‘eisaR’

January 26, 2021

Title Exon-Intron Split Analysis (EISA) in R

Version 1.3.0

Description Exon-intron split analysis (EISA) uses ordinary RNA-seq data to measure changes in mature RNA and pre-mRNA reads across different experimental conditions to quantify transcriptional and post-transcriptional regulation of gene expression. For details see Gaidatzis et al., Nat Biotechnol 2015. doi: 10.1038/nbt.3269. eisaR implements the major steps of EISA in R.

Depends R (>= 4.0.0)

License GPL-3

biocViews Transcription, GeneExpression, GeneRegulation,
FunctionalGenomics, Transcriptomics, Regression, RNASeq

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat, BiocStyle, QuasR, Rbowtie,
Biostrings, BSgenome, BSgenome.Hsapiens.UCSC.hg38, ensemblDb

VignetteBuilder knitr

Imports graphics, stats, GenomicRanges, GenomicFeatures, S4Vectors,
IRanges, AnnotationDbi, limma, edgeR, methods,
SummarizedExperiment, BiocGenerics, rtracklayer, utils

URL <https://github.com/fmicompbio/eisaR>

BugReports <https://github.com/fmicompbio/eisaR/issues>

git_url <https://git.bioconductor.org/packages/eisaR>

git_branch master

git_last_commit d864d3f

git_last_commit_date 2020-10-27

Date/Publication 2021-01-25

Author Michael Stadler [aut, cre],
Dimos Gaidatzis [aut],
Lukas Burger [aut],
Charlotte Soneson [aut]

Maintainer Michael Stadler <michael.stadler@fmi.ch>

R topics documented:

exportToGtf	2
getFeatureRanges	3
getRegionsFromTxDb	4
getTx2Gene	5
plotEISA	6
runEISA	7

Index	11
--------------	-----------

exportToGtf	<i>Export GRangesList to GTF</i>
-------------	----------------------------------

Description

Export the features in a GRangesList generated by getFeatureRanges to a GTF file. The function will represent each row of each of the entries as an "exon", each individual entry as a "transcript", and aggregate all features belonging to the same gene as a "gene" entry in the GTF file.

Usage

```
exportToGtf(grl, filepath)
```

Arguments

grl	GRangesList object, typically generated by getFeatureRanges
filepath	Path to output GTF file

Value

Does not return anything, generates a GTF file

Author(s)

Charlotte Soneson

Examples

```
## Get feature ranges
grl <- getFeatureRanges(
  gtf = system.file("extdata/small_example.gtf", package = "eisaR"),
  featureType = c("spliced", "intron"),
  intronType = "separate",
  flankLength = 5L,
  joinOverlappingIntrons = FALSE,
  verbose = TRUE
)

## Export GTF
exportToGtf(grl = grl, filepath = file.path(tempdir(), "exported.gtf"))
```

getFeatureRanges	<i>Generate a GRangesList object with feature ranges</i>
------------------	--

Description

Generate a GRangesList object with genomic ranges for (any combination of) spliced transcripts, unspliced transcripts and introns.

Usage

```
getFeatureRanges(  
  gtf,  
  featureType = c("spliced", "intron"),  
  intronType = "separate",  
  flankLength = 90L,  
  joinOverlappingIntrons = FALSE,  
  verbose = TRUE  
)
```

Arguments

gtf	Path to gtf file.
featureType	Character vector indicating the type(s) of features to extract, any subset of c("spliced", "intron", "
intronType	Character vector indicating how to define the introns (only used if "intron" is part of featureType). Has to be either "separate" (introns are defined for each transcript separately) or "collapse" (transcripts of the same gene are first collapsed before introns are defined as any non-exonic part of the gene locus).
flankLength	Integer scalar indicating the length of the flanking sequence added to each side of each extracted intron (only used if "intron" is included in featureType).
joinOverlappingIntrons	Logical scalar indicating whether two introns that overlap (after adding the flanking sequences) should be joined into one feature.
verbose	Logical scalar, whether to print out progress messages.

Value

Returns a GRangesList object where each element represents one extracted feature. The metadata of this object contains two data.frames mapping corresponding identifiers between the different feature types, as well as a list of all features for each type.

Author(s)

Charlotte Sonesson

Examples

```
## Get feature ranges  
grl <- getFeatureRanges(  
  gtf = system.file("extdata/small_example.gtf", package = "eisaR"),  
  featureType = c("spliced", "intron"),
```

```

    intronType = "separate",
    flankLength = 5L,
    joinOverlappingIntrons = FALSE,
    verbose = TRUE
  )

  ## GRangesList
  grl

  ## Corresponding transcript/gene IDs
  S4Vectors::metadata(grl)$corrtx
  S4Vectors::metadata(grl)$corrgene

  ## List of features of different types
  S4Vectors::metadata(grl)$featurelist

  ## Get feature sequences
  if (requireNamespace("BSgenome", quietly = TRUE)) {
    library(BSgenome)
    genome <- Biostrings::readDNAStringSet(
      system.file("extdata/small_example_genome.fa", package = "eisaR"))
    seqs <- GenomicFeatures::extractTranscriptSeqs(x = genome,
                                                    transcripts = grl)

    seqs
  }

```

getRegionsFromTxDb *Get exonic/gene body regions from a transcript database.*

Description

From a transcript database package ([TxDb](#)), extract exonic and gene body ranges for use with EISA. These regions can be used to quantify RNA-seq alignments in exons and gene bodies, respectively. Intronic counts can then be obtained from the difference between gene bodies and exonic region counts.

Usage

```
getRegionsFromTxDb(txdb, exonExt = 10L, strandedData = TRUE)
```

Arguments

txdb	a TxDb or an EnsDb object with the transcript annotations.
exonExt	numeric (default = 10L). Exonic ranges will be extended on either side by this many nucleotides, in order to avoid "bleed-over" of exonic alignments into adjacent intronic regions.
strandedData	logical(1). If TRUE, the RNA-seq data is assumed to be strand-specific, and therefore only overlapping genes that are on the same strand will be filtered out. If FALSE, also genes overlapping on opposite strands will be filtered out.

Details

The exonic regions are generated as follows:

1. extract exons by gene from the txdb
2. extend each exon by exonExt
3. combine overlapping exons within each gene
4. create gene body ranges from the most extreme exonic coordinates
5. filter out genes that have only a single exon (no intron), have exons on more than a single chromosome or on both strands, or that overlap other genes

Value

a list with elements "exons" and "genebodies", containing named GenomicRanges objects with ranges for exons and gene bodies, respectively.

Author(s)

Michael Stadler

See Also

[TxDb](#) for details on TxDb objects and how to create them, e.g. from .gtf files.

Examples

```
txdb <- AnnotationDbi::loadDb(system.file("extdata", "hg19sub.sqlite", package = "eisaR"))
regL <- getRegionsFromTxDb(txdb)
lengths(regL)
```

getTx2Gene

Generate a transcript-to-gene mapping from a GRangesList

Description

Generate a data.frame mapping transcript IDs to gene IDs, based on a GRangesList object generated by getFeatureRanges.

Usage

```
getTx2Gene(grl, filepath = NULL)
```

Arguments

grl	GRangesList object, typically generated by getFeatureRanges
filepath	Either NULL or the path to a file where the transcript-to-gene mapping will be written

Value

Invisibly returns a data.frame with the transcript-to-gene mapping.

Author(s)

Charlotte Soneson

Examples

```
## Get feature ranges
gr1 <- getFeatureRanges(
  gtf = system.file("extdata/small_example.gtf", package = "eisaR"),
  featureType = c("spliced", "intron"),
  intronType = "separate",
  flankLength = 5L,
  joinOverlappingIntrons = FALSE,
  verbose = TRUE
)

## Get transcript-to-gene mapping
t2g <- getTx2Gene(gr1 = gr1)
t2g
```

plotEISA

*Visualize the results from an exon-intron split analysis.***Description**

plotEISA takes the return value from [runEISA](#) and generates a scatterplot of intronic versus exonic changes.

Usage

```
plotEISA(
  x,
  contrast = c("ExIn", "none"),
  minLfc = NULL,
  maxFDR = 0.05,
  genecolors = c("#E41A1C", "#497AB3", "#222222"),
  ...
)
```

Arguments

x	list with EISA results, typically the return value from runEISA
contrast	one of "ExIn" or "none". If "ExIn" (the default), genes that significantly differ between exonic and intronic changes are highlighted. "none" turns off gene highlighting.
minLfc	NULL or numeric(1) with the minimal absolute log2 fold change to color a gene. If NULL (the default), no fold changes are not used to select genes for highlighting.
maxFDR	numeric(1) with maximal false discovery rate for gene highlighting.
genecolors	Vector of length three specifying the colors to use for genes that are significantly up, down or unchanged.

... further arguments past to plot(). Parameters that will be set automatically unless given in the arguments are:

- pch** : plot symbol (default: ".")
- cex** : plot symbol expansion factor (default: 2)
- col** : plot symbol color (default: according to contrast and genecolors)
- xlab/ylab** : axis labels

Value

NULL (invisibly)

Author(s)

Michael Stadler

Examples

```
# see the help for runEISA() for a full example
```

runEISA

Run Exon-Intron Split Analysis.

Description

Starting from count tables with exonic and intronic counts for two conditions, perform all the steps in EISA (normalize, identify quantifiable genes, calculate contrasts and their significance).

Usage

```
runEISA(  
  cntEx,  
  cntIn,  
  cond,  
  method = NULL,  
  modelSamples = TRUE,  
  geneSelection = c("filterByExpr", "none", "Gaidatzis2015"),  
  statFramework = c("QLF", "LRT"),  
  effects = c("predFC", "Gaidatzis2015"),  
  psct = 2,  
  sizeFactor = c("exon", "intron", "individual"),  
  recalcNormFactAfterFilt = TRUE,  
  recalLibSizeAfterFilt = FALSE,  
  ...  
)
```

Arguments

cntEx	Gene by sample matrix with exonic counts, OR a SummarizedExperiment with two assays named exon and intron, containing exonic and intronic counts, respectively. If cntEx is a SummarizedExperiment, cntIn will be disregarded.
cntIn	Gene by sample matrix with intronic counts. Must have the same structure as cntEx (same number and order of rows and columns) if cntEx is a matrix. Will be disregarded if cntEx is a SummarizedExperiment.
cond	numeric, character or factor with two levels that groups the samples (columns of cntEx and cntIn) into two conditions. The contrast will be defined as secondLevel - firstLevel.
method	One of NULL (the default) or "Gaidatzis2015". If "Gaidatzis2015", gene filtering, statistical analysis and calculation of contrasts is performed as described in Gaidatzis et al. 2015, and the statistical analysis is based on <code>glmFit</code> and <code>glmLRT</code> . This is done by setting the arguments <code>modelSamples</code> , <code>geneSelection</code> , <code>effects</code> , <code>pscmt</code> , <code>statFramework</code> , <code>sizeFactor</code> , <code>recalcNormFactAfterFilt</code> and <code>recalcLibSizeAfterFilt</code> to appropriate values (see details), overriding the defaults or any value passed to these arguments. If NULL, the default values of the arguments will be used instead (recommended).
modelSamples	Whether to include a sample identifier in the design matrix of the statistical model. If TRUE, potential sample effects that affect both exonic and intronic counts of that sample will be taken into account, which could result in higher sensitivity (default: TRUE).
geneSelection	Controls how to select quantifiable genes. One of the following: <p>"filterByExpr": (default) First, counts are normalized using <code>calcNormFactors</code>, treating intronic and exonic counts as individual samples. Then, <code>filterByExpr</code> is used with default parameters to select quantifiable genes.</p> <p>"none": This will use all the genes provided in the count tables, assuming that an appropriate selection of quantifiable genes has already been done.</p> <p>"Gaidatzis2015": First, intronic and exonic counts are linearly scaled to the mean library size (estimated as the sum of all intronic or exonic counts, respectively). Then, quantifiable genes are selected as the genes with counts x that fulfill $\log_2(x + 8) > 5$ in both exons and introns.</p>
statFramework	Selects the framework within edgeR that is used for the statistical analysis. One of: <p>"QLF": (default) Quasi-likelihood F-test using <code>glmQLFit</code> and <code>glmQLFTest</code>. This framework is highly recommended as it gives stricter error rate control by accounting for the uncertainty in dispersion estimation.</p> <p>"LRT": Likelihood ratio test using <code>glmFit</code> and <code>glmLRT</code>.</p>
effects	How the effects (contrasts or log2 fold-changes) are calculated. One of: <p>"predFC": (default) Fold-changes are calculated using the fitted model with <code>predFC</code> with <code>prior.count = pscmt</code>. Please note that if a sample factor is included in the model (<code>modelSamples=TRUE</code>), effects cannot be obtained from that model. In that case, effects are obtained from a simpler model without sample effects.</p> <p>"Gaidatzis2015": Fold-changes are calculated using the formula $\log_2((x + pscmt)/(y + pscmt))$. If <code>pscmt</code> is not set to 8, runEISA will warn that this deviates from the method used in Gaidatzis et al., 2015.</p>

<code>pscnt</code>	numeric(1) with pseudocount to add to read counts (default: 2). For method = "Gaidatzis2015", it is set to 8. It is added to scaled read counts used in <code>geneSelection = "Gaidatzis2015"</code> and <code>effects = "Gaidatzis2015"</code> , or else used in <code>cpm(..., prior.count = pscnt)</code> and <code>predFC(..., prior.count = pscnt)</code> .
<code>sizeFactor</code>	How the size factors are calculated in the analysis. If 'exon' (default), the exon-derived size factors are used also for the columns corresponding to intronic counts. If 'intron', the intron-derived size factors are used also for the columns corresponding to exonic counts. If 'individual', column-wise size factors are calculated.
<code>recalcNormFactAfterFilt</code>	Logical, indicating whether normalization factors should be recalculated after filtering out lowly expressed genes.
<code>recalcLibSizeAfterFilt</code>	Logical, indicating whether library sizes should be recalculated after filtering out lowly expressed genes.
<code>...</code>	additional arguments passed to the <code>DGEList</code> constructor, such as <code>lib.size</code> or <code>genes</code> .

Details

Setting `method = "Gaidatzis2015"` has precedence over other argument values and corresponds to setting: `modelSamples = FALSE`, `geneSelection = "Gaidatzis2015"`, `statFramework = "LRT"`, `effects = "Gaidatzis2015"`, `pscnt = 8`, `sizeFactor = "individual"`, `recalcNormFactAfterFilt = TRUE`, `recalcLibSizeAfterFilt = FALSE`.

Value

a list with elements

fracIn fraction intronic counts in each sample

contrastName contrast name

contrasts contrast matrix for quantifiable genes, with average log₂ fold-changes in exons (Dex), in introns (Din), and average difference between log₂ fold-changes in exons and introns (Dex.Din)

DGEList `DGEList` object used in model fitting

tab.ExIn statistical results for differential changes between exonic and intronic contrast, an indication for post-transcriptional regulation.

contr.ExIn contrast vector used for testing the difference between exonic and intronic contrast (results in `tab.ExIn`)

designMatrix design matrix used for testing the difference between exonic and intronic contrast (results in `tab.ExIn`)

params a list with parameter values used to run EISA

Author(s)

Michael Stadler

References

Analysis of intronic and exonic reads in RNA-seq data characterizes transcriptional and post-transcriptional regulation. Dimos Gaidatzis, Lukas Burger, Maria Florescu and Michael B. Stadler Nature Biotechnology, 2015 Jul;33(7):722-9. doi: 10.1038/nbt.3269.

See Also

[DGEList](#) for DGEList object construction, [calcNormFactors](#) for normalization, [filterByExpr](#) for gene selection, [glmFit](#) and [glmQLFit](#) for statistical analysis.

Examples

```
cntEx <- readRDS(system.file("extdata", "Fig3abc_GSE33252_rawcounts_exonic.rds",
                             package = "eisaR"))[, -1]
cntIn <- readRDS(system.file("extdata", "Fig3abc_GSE33252_rawcounts_intronic.rds",
                             package = "eisaR"))[, -1]
cond <- factor(c("ES", "ES", "TN", "TN"))
res <- runEISA(cntEx, cntIn, cond)
plotEISA(res)
```

Index

`calcNormFactors`, [8](#), [10](#)

`DGEList`, [9](#), [10](#)

`exportToGtf`, [2](#)

`filterByExpr`, [8](#), [10](#)

`getFeatureRanges`, [3](#)

`getRegionsFromTxDb`, [4](#)

`getTx2Gene`, [5](#)

`glmFit`, [8](#), [10](#)

`glmLRT`, [8](#)

`glmQLFit`, [8](#), [10](#)

`glmQLFTest`, [8](#)

`plotEISA`, [6](#)

`predFC`, [8](#)

`runEISA`, [6](#), [7](#)

`TxDb`, [4](#), [5](#)