Package 'JunctionSeq'

April 23, 2015

Version 0.3.5

Date 2015-04-23

Title JunctionSeq: A Utility for Detection of Differential Splice-Site Usage in RNA-Seq data

Author Stephen Hartley, PhD

Maintainer Stephen Hartley <stephen.hartley@nih.gov>

Depends R (>= 3.0.2), statmod, plotrix, methods, stringr, Biobase

Suggests MASS, Cairo, RSvgDevice, knitr

Enhances parallel, BiocParallel

Description Utility for Detection of Differential Splice-Site Usage in RNA-Seq data.

License file LICENSE

VignetteBuilder knitr

URL https://GitHub.com/hartleys/JunctionSeq

BugReports https://GitHub.com/hartleys/JunctionSeq

R topics documented:

buildAllPlots 2
buildAllPlotsForGene 5
estimateEffectSizes 8
estimateJunctionSeqDispersions 9
estimateSizeFactors 10
fitDispersionFunction 10
generateAllExpressionEstimates 11
generateCompleteResults 12
JunctionSeqCountSet-class 13
plotDispEsts 14
plotJunctionSeqResultsForGene 15
plotMA 17
readAnnotationData 19
readJunctionSeqCounts 19
runJunctionSeqAnalyses 20
testJunctionsForDJU 22
writeBedTrack 23
writeCompleteResults 25

Index 27

`buildAllPlots` *Creating result plots*

Description

Saves a large battery of plots displaying the analysis results, for the purposes of data visualization.

Usage

```
buildAllPlots(jscs,
              outfile.prefix = "./",
              flat.gff.data = NULL, flat.gff.file = NULL,
              gene.list = NULL, FDR.threshold = 0.01,
              use.plotting.device = "png",
              use.vst=FALSE,use.log = TRUE, truncateBelowOne = TRUE,
              exon.rescale.factor = 0.3,
              ma.plot=FALSE, variance.plot=FALSE,
              with.TX=TRUE,without.TX=TRUE,
              expr.plot=TRUE,normCounts.plot=TRUE,
              rExpr.plot=TRUE,rawCounts.plot=TRUE,
              colorRed.FDR.threshold = FDR.threshold,
              color=NULL,
              plot.exon.results = FALSE,
              plot.splice.results = TRUE,
              plot.novel.splice.results = plot.splice.results,
              plot.untestable.results = FALSE,
              plot.lwd=3, axes.lwd = plot.lwd, anno.lwd = plot.lwd,
              par.cex = 1, anno.cex.text = 2, anno.cex.axis = anno.cex.text, a
              drawCoordinates = TRUE,
              arrows.length = 0.25,
              base.plot.height = 22.222, base.plot.width = 22.222,
              base.plot.units = "in",
              plotting.device.params = list(pointsize = 18, res = 72),
              number.plots = TRUE,
              openPlottingDeviceFunc, closePlottingDeviceFunc,
              verbose=TRUE,
              ...)
```

Arguments

`jscs`	A `JunctionSeqCountSet`. Usually created by `runJunctionSeqAnalyses`. Alternatively, this can be created manually by `readJunctionSeqCounts`. However in this case a number of additional steps will be necessary: Dispersions and size factors must then be set, usually using functions `estimateSizeFactors` and `estimateJunctionSeqDispersions`. Hypothesis tests must be performed by `testJunctionsForDJU`. Effect sizes and parameter estimates must be created via `estimateEffectSizes`.
`outfile.prefix`	The prefix file path to save the images to.

`flat.gff.data`
: A data frame containing the annotation data. This can be extracted from a flat gff file using the `readAnnotationData` function. If multiple plots are going to be generated it may be preferable to use this to save runtime.

`flat.gff.file`
: A flattened gff file containing the annotation data. Only one of flat.gff.data or flat.gff.file should be set.

`gene.list`
: Character vector. List of genes to plot. Either this variable OR FDR.threshold must be set.

`FDR.threshold`
: If this option is used, genes will be selected for plotting based on the presence of statistically significant junctions. The adjusted-p-value threshold used to determine significance. Only genes containing at least 1 significant feature will be plotted.

`use.plotting.device`
: The plotting device to use.

`use.vst`
: Logical. If TRUE, all plots will be scaled via a variance stabilizing transform.

`use.log`
: Logical. If TRUE, all plots will be log-scaled.

`truncateBelowOne`
: Logical. If TRUE, all values below 1 will be linear-scaled. If use.log is FALSE, this does nothing.

`exon.rescale.factor`
: Floating point numeric value. To improve readability the exons drawn in the coordinate annotation are rescaled by default so that they take up 30 percent of the x axis. This makes the plots easier to read, as exons are usually much smaller than introns and thus a group of clustered exons can be hard to distinguish when plotted on a simple scale. If this value is set to NA then the exons and introns will be drawn on the same linear scale.

`ma.plot`
: if TRUE, generate and save a MA plot. A MA-plot is a plot of fold change versus base mean normalized counts.

`variance.plot`
: if TRUE, generate and save a plot of the dispersion as a function of the base mean.

`with.TX`
: if TRUE, save expression plots with the full transcripts printed

`without.TX`
: if TRUE, save expression plots with only the compiled exons printed. Note that if this and with.TX.plot are both TRUE, both versions will be saved seperately.

`expr.plot`
: if TRUE, save an expression plot of the expression parameter estimates for each splice site, for each condition.

`normCounts.plot`
: if TRUE, save an expression plot of the normalized mean counts for each splice site, for each sample.

`rExpr.plot`
: if TRUE, save an expression plot of the expression parameter estimates, relative to gene-wide expression, for each splice site, for each condition.

`rawCounts.plot`
: if TRUE, save an expression plot of the raw counts for each splice site, for each sample. Note that these will never be VST-transformed, even when use.vst == TRUE.

`colorRed.FDR.threshold`
: The adjusted-p-value threshold used to determine whether a feature should be marked as "significant" and colored pink. By default this will be the same as the FDR.threshold.

`color` A vector of R colors, named for each possible value of condition. By default, it will attempt to choose reasonable colors for each condition.

`plot.exon.results` Logical. If TRUE, plot results for exons. Technically speaking, JunctionSeq can be used to do DEXSeq-style analyses on exon partitions. However this functionality is for advanced users only.

`plot.splice.results` Logical. If TRUE, plot results for splice junctions. For advanced users only.

`plot.novel.splice.results` Logical. If TRUE, plot results for novel splice junctions. If false, novel splice junctions will be ignored.

`plot.untestable.results` Logical. If TRUE, plots splice junctions that had coverage that was too low to be tested.

`plot.lwd` the line width for the plotting lines.

`axes.lwd` the line width for the axes.

`anno.lwd` the line width for the various other annotation lines.

`par.cex` The base cex value to be passed to par() immediately before all plots are created. See `par`.

`anno.cex.text` The font size multiplier for most annotation text. This will be multiplied by a factor of the par.cex value. More specifically: The cex value to be passed to all function calls that take graphical parameters. See `par`.

`anno.cex.axis` The font size multiplier for the axis text. This will be multiplied by a factor of the par.cex value. More specifically: The cex.axis value to be passed to all function calls that take graphical parameters. See `par`.

`anno.cex.main` The font size multiplier for the main title text. This will be multiplied by a factor of the par.cex value. More specifically: The cex.main value to be passed to all function calls that take graphical parameters. See `par`.

`drawCoordinates` Whether to label the genomic coordinates at the bottom of the plot.

`arrows.length` The length (in inches), of the arrows that indicate gene strandedness.

`base.plot.height` The base height of the standard-sized plots. The plots that include the transcript set will be 1.5x this height.

`base.plot.width` The width of the plots.

`base.plot.units` The units of measurement for the plot height and width. Default is px, or pixels.

`plotting.device.params` Additional parameters to be passed to the plotting device.

`number.plots` Whether to number each gene in the image names, based on either the order they appear in the input gene.list, or in order of ascending p-values.

`openPlottingDeviceFunc` An R function. This option can be used to use plotting devices other than the ones directly supported by JunctionSeq. This must be a function that must have

3 parameters: filename, heightMult, and widthMult. It should open the desired plotting device. For advanced users only.

`closePlottingDeviceFunc`
: An R function. This must be used in conjunction with openPlottingDeviceFunc. For most devices, you can just use the function "dev.off". For advanced users only.

`verbose`
: if TRUE, send debugging and progress messages to the console / stdout.

`...`
: Additional options to pass to plotting functions, particularly graphical parameters.

`buildAllPlotsForGene` *Creating result plots*

Description

Saves a large battery of plots displaying the analysis results, for the purposes of data visualization.

Usage

```
buildAllPlotsForGene(geneID, jscs,
                     outfile.prefix = "./",
                     flat.gff.data = NULL, flat.gff.file = NULL,
                     use.plotting.device = "png",
                     use.vst=FALSE, use.log = TRUE, truncateBelowOne = TRUE,
                     exon.rescale.factor = 0.3,
                     with.TX=TRUE,without.TX=TRUE,
                     expr.plot=TRUE,normCounts.plot=TRUE,
                     rExpr.plot=TRUE,rawCounts.plot=TRUE,
                     colorRed.FDR.threshold = 0.01,
                     color=NULL,
                     plot.exon.results = FALSE,
                     plot.splice.results = TRUE,
                     plot.novel.splice.results = plot.splice.results,
                     plot.untestable.results = FALSE,
                     plot.lwd=3, axes.lwd = plot.lwd, anno.lwd = plot.lwd,
                     par.cex = 1, anno.cex.text = 2,
                     anno.cex.axis = anno.cex.text, anno.cex.main = anno.cex.tex
                     drawCoordinates = TRUE,
                     arrows.length = 0.25,
                     base.plot.height = 22.222, base.plot.width = 22.222,
                     base.plot.units = "in",
                     plotting.device.params = list(pointsize = 18, res = 72),
                     openPlottingDeviceFunc = NULL, closePlottingDeviceFunc = NU
                     verbose=TRUE,  ...)
```

Arguments

geneID	Character string. Which gene to plot.
jscs	A JunctionSeqCountSet. Usually created by runJunctionSeqAnalyses. Alternatively, this can be created manually by readJunctionSeqCounts. However in this case a number of additional steps will be necessary: Dispersions and size factors must then be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions. Hypothesis tests must be performed by testJunctionsForDJU. Effect sizes and parameter estimates must be created via estimateEffectSizes.
outfile.prefix	The prefix file path to save the images to.
flat.gff.data	A data frame containing the annotation data. This can be extracted from a flat gff file using the readAnnotationData function. If multiple plots are going to be generated it may be preferable to use this to save runtime.
flat.gff.file	A flattened gff file containing the annotation data. Only one of flat.gff.data or flat.gff.file should be set.
use.plotting.device	The plotting device to use.
use.vst	Logical. If TRUE, all plots will be scaled via a variance stabilizing transform.
use.log	Logical. If TRUE, all plots will be log-scaled.
truncateBelowOne	Logical. If TRUE, all values below 1 will be linear-scaled. If use.log is FALSE, this does nothing.
exon.rescale.factor	Numeric. Exons will be proportionately scaled-up so that the exonic regions make up this fraction of the horizontal plotting area. If negative, exons and introns will be plotted to a common scale.
with.TX	if TRUE, save expression plots with the full transcripts printed
without.TX	if TRUE, save expression plots with only the compiled exons printed. Note that if this and with.TX.plot are both TRUE, both versions will be saved seperately.
expr.plot	if TRUE, save an expression plot of the expression parameter estimates for each splice site, for each condition.
normCounts.plot	if TRUE, save an expression plot of the normalized mean counts for each splice site, for each sample.
rExpr.plot	if TRUE, save an expression plot of the expression parameter estimates, relative to gene-wide expression, for each splice site, for each condition.
rawCounts.plot	if TRUE, save an expression plot of the raw counts for each splice site, for each sample. Note that these will never be VST-transformed, even when use.vst == TRUE.
colorRed.FDR.threshold	The adjusted-p-value threshold used to determine whether a feature should be marked as "significant" and colored pink. By default this will be the same as the FDR.threshold.
color	A vector of R colors, named for each possible value of condition. By default, it will attempt to choose reasonable colors for each condition.

`plot.exon.results`
: Logical. If TRUE, plot results for exons. Technically speaking, JunctionSeq can be used to do DEXSeq-style analyses on exon partitions. However this functionality is for advanced users only.

`plot.splice.results`
: Logical. If TRUE, plot results for splice junctions. For advanced users only.

`plot.novel.splice.results`
: Logical. If TRUE, plot results for novel splice junctions. If false, novel splice junctions will be ignored.

`plot.untestable.results`
: Logical. If TRUE, plots splice junctions that had coverage that was too low to be tested.

`plot.lwd`
: the line width for the plotting lines.

`axes.lwd`
: the line width for the axes.

`anno.lwd`
: the line width for the various other annotation lines.

`par.cex`
: The base cex value to be passed to par() immediately before all plots are created. See `par`.

`anno.cex.text`
: The font size multiplier for most annotation text. This will be multiplied by a factor of the par.cex value. More specifically: The cex value to be passed to all function calls that take graphical parameters. See `par`.

`anno.cex.axis`
: The font size multiplier for the axis text. This will be multiplied by a factor of the par.cex value. More specifically: The cex.axis value to be passed to all function calls that take graphical parameters. See `par`.

`anno.cex.main`
: The font size multiplier for the main title text. This will be multiplied by a factor of the par.cex value. More specifically: The cex.main value to be passed to all function calls that take graphical parameters. See `par`.

`drawCoordinates`
: Whether to label the genomic coordinates at the bottom of the plot.

`arrows.length`
: The length (in inches), of the arrows that indicate gene strandedness.

`base.plot.height`
: The base height of the standard-sized plots. The plots that include the transcript set will be 1.5x this height.

`base.plot.width`
: The width of the plots.

`base.plot.units`
: The units of measurement for the plot height and width. Default is px, or pixels.

`plotting.device.params`
: Additional parameters to be passed to the plotting device.

`openPlottingDeviceFunc`
: An R function. This option can be used to use plotting devices other than the ones directly supported by JunctionSeq. This must be a function that must have 3 parameters: filename, heightMult, and widthMult. It should open the desired plotting device. For advanced users only.

`closePlottingDeviceFunc`
: An R function. This must be used in conjunction with openPlottingDeviceFunc. For most devices, you can just use the function "dev.off". For advanced users only.

verbose	if TRUE, send debugging and progress messages to the console / stdout.
...	Additional options to pass to plotting functions, particularly graphical parameters.

`estimateEffectSizes` *Estimate Effect Factors*

Description

This function runs fits another generalized linear model to the data, this one intended for use in estimating the effect sizes and expression estimates.

For most purposes this function should not be needed, but might be useful to advanced users performing non-standard analyses. This and all other standard analysis functions can be run automatically via the `runJunctionSeqAnalyses` function.

Usage

```
estimateEffectSizes(jscs,
     effect.formula = formula(~ condition + countbin + condition : countbin),
     nCores=1,
     dispColumn="dispersion",
     verbose = TRUE)
```

Arguments

jscs	A `JunctionSeqCountSet`. Usually initially created by `readJunctionSeqCounts`. Size factors must be set, usually using functions `estimateSizeFactors` and `estimateJunctionSeqDispersions`.
effect.formula	For advanced users. The base formula for the model used for effect size estimation. NOTE: the biological condition to be tested must be named "condition".
nCores	The number of cores to use. Note that multicore functionality may not be available on all platforms. JunctionSeq attempts to use the BiocParallell package if it can be found installed. Otherwise it will attempt to fallback to the multicore package. If neither package can be found it will fallback to single core execution.
dispColumn	Character value. The name of the `fData(jscs)` column in which the model dispersion is stored.
verbose	if TRUE, send debugging and progress messages to the console / stdout.

Value

A JunctionSeqCountSet, with effect size results included.

`estimateJunctionSeqDispersions` *JunctionSeq Dispersion Estimation*

Description

This method estimates the sample dispersion for each counting bin (in other words, each splice junction locus).

For most purposes this function should not be needed, but might be useful to advanced users performing non-standard analyses. This and all other standard analysis functions can be run automatically via the `runJunctionSeqAnalyses` function.

Usage

```
estimateJunctionSeqDispersions(jscs,
        test.formula1 = formula(~ sample + countbin + condition : countbin),
        minCount=10, nCores=1,
        test.aggregated.genes = FALSE,
        use.alternate.method  = TRUE,
        verbose = TRUE);
```

Arguments

`jscs`	A `JunctionSeqCountSet`. Usually initially created by `readJunctionSeqCounts`. Size factors must be set, usually using functions `estimateSizeFactors` and `estimateJunctionSeqDispersions`.
`test.formula1`	The model formula. Note that this formula is different from the formula used to calculate parameter estimates and effect size. This is because the two noise components (gene-level and countbin-level noise) are folded into the sample term. Since we only intend to test the condition-countbin interaction, we do not need to model the gene-level differential expression. NOTE: the biological condition to be tested MUST be named "condition".
`minCount`	Integer value. The minimum mean normalized read count threshold required. Default is 10, which is reasonable for most purposes.
`nCores`	The number of cores to use. Note that multicore functionality may not be available on all platforms.
`test.aggregated.genes`	Logical value. Whether to attempt to test "aggregate genes" which consist of multiple genes that overlap with one another. Note that inclusion of aggregate genes may affect the false discovery rate, since by their very nature aggregate genes will often show differential splice junction usage, as the two genes will often be regulated independently.
`use.alternate.method`	ADVANCED USERS ONLY: Logical value. Determines whether to use the JunctionSeq standard model framework, or the DEXSeq model framework.
`verbose`	A boolean flag indicating whether or not to print progress information during execution. (Default=FALSE)

Value

A JunctionSeqCountSet, with dispersion results included.

`estimateSizeFactors`
Estimate Size Factors

Description

Estimate size factors, which are scaling factors used as "offsets" by the statistical model to make the different samples comparable. This is necessary because the different samples may have been sequenced to slightly different depths. Additionally, the presence of differentially expressed genes may cause the apparent depth of many genes to appear different.

This function uses the "geometric" size factor normalization method, which is identical to the one used by DESeq, DESeq2, DEXSeq, and the default method used by CuffDiff.

For most purposes this function should not be needed, but might be useful to advanced users performing non-standard analyses. This and all other standard analysis functions can be run automatically via the `runJunctionSeqAnalyses` function.

Usage

```
estimateSizeFactors(jscs);
```

Arguments

`jscs`	A `JunctionSeqCountSet`. Usually initially created by `readJunctionSeqCounts`. Size factors must be set, usually using functions `estimateSizeFactors` and `estimateJunctionSeqDispersions`.

Value

A JunctionSeqCountSet, with size factors included.

`fitDispersionFunction`
Fit Shared Dispersion Function

Description

Fit dispersion function to share dispersion information between features across the genome.

For most purposes this function should not be needed, but might be useful to advanced users performing non-standard analyses. This and all other standard analysis functions can be run automatically via the `runJunctionSeqAnalyses` function.

Usage

```
fitDispersionFunction(jscs, verbose);
```

Arguments

jscs	A JunctionSeqCountSet. Usually initially created by readJunctionSeqCounts. Size factors must be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions.
verbose	if TRUE, send debugging and progress messages to the console / stdout.

Value

A JunctionSeqCountSet, with dispersion results included.

---

generateAllExpressionEstimates

TODO

Description

ADVANCED USERS ONLY: This function is intended for advanced users only. For general use, using the generateCompleteResults function is recommended, which will in turn call this function internally.

Usage

```
generateAllExpressionEstimates(jscs,
                               nCores = 1,
                               fitExpToVar="condition",
                               verbose=TRUE)
```

Arguments

jscs	A JunctionSeqCountSet. Usually created by runJunctionSeqAnalyses. Alternatively, this can be created manually by readJunctionSeqCounts. Dispersions and size factors must then be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions. Hypothesis tests must be performed by testJunctionsForDJU.
nCores	The number of cores to use. Note that multicore functionality may not be available on all platforms.
fitExpToVar	Character value. The biological condition to fit the model to.
verbose	Logical value. Determines how much debugging into to message to the console.

`generateCompleteResults` *generateCompleteResults (DEPRECIATED)*

Description

This function is DEPRECIATED. Use writeCompleteResults instead.

This function takes the raw DEXSeq results and merges in feature annotations, as well as calculating and merging in a number of different normalized and fitted values for each level of the condition variable.

Usage

```
generateCompleteResults(jscs, outfile.prefix = NULL,
                        verbose = TRUE,nCores=1,
                        gzip.output = TRUE)
```

Arguments

`jscs`	A `JunctionSeqCountSet`. Usually created by `runJunctionSeqAnalyses`. Alternatively, this can be created manually by `readJunctionSeqCounts`. Dispersions and size factors must then be set, usually using functions `estimateSizeFactors` and `estimateJunctionSeqDispersions`. Hypothesis tests must be performed by `testJunctionsForDJU`.
`outfile.prefix`	A string indicating the filename prefix where output files should be saved. If NULL, no data will be saved to file. (Default=NULL)
`verbose`	A boolean flag indicating whether or not to print progress information during execution. (Default=TRUE)
`nCores`	If the package multicore is installed, the number of cores to use. If multicore is not installed, then setting this option anything other than the default of 1 does nothing but produce a warning.
`gzip.output`	A boolean indicating whether output files should be gzipped.

Value

A data frame containing the results for all listed splice sites of genes for which the GLM converged successfully.

Each row is a splice site. The columns are:

- First, All columns from the res input file.
- exprVST_conditionVal: the variance-stabilizing-transformed fitted expression estimates for the splice site at each condition value.
- expr_conditionVal: the un-transformed fitted expression estimates for the splice site at each condition value.
- rExprVST_conditionVal: the variance-stabilizing-transformed fitted relative expression estimates for the splice site at each condition value.
- rExpr_conditionVal: as above, but without the Variance-stabilizing transformation.

- normCountVST_sampleID: Variance-stabilizing-transformed normalized counts for the splice site at each sample.
- normCount_sampleID: Variance-stabilizing-transformed normalized counts for the splice site at each sample.
- Finally, (if anno.file is non-null), the columns from the annotation file, matched to the given genes/splice sites.

Assuming the annotation file is the standard tab-delimited annotation file generated by prepare_annotation_with_splices.p
the final columns will be:

- featureType: the type of the feature (for these analyses, always equal to either "splice_site" or "novel_splice_site".
- chrom: The chromosome position.
- start
- end
- strand: '+' or '-'
- gene_id
- part_number: the 3-digit sub-feature ID
- transcripts: the list of transcripts, delimited by '+'

`JunctionSeqCountSet-class`
JunctionSeqCountSet object and constructors

Description

The `JunctionSeqCountSet` is a subclass of the bioconductor class `eSet`, designed to contain all data related to a JunctionSeq analysis.

Usage

```
newJunctionSeqCountSet( countData,
                        geneCountData,
                        design,
                        geneIDs,
                        countbinIDs,
                        featureIntervals=NULL,
                        transcripts=NULL)
```

Arguments

`countData`	A matrix of junction-level count data of non-negative integer values. The rows correspond to counts for each splice-junction counting bin, the columns correspond to samples. Note that biological replicates should each get their own column, while the counts of technical replicates (i.e., several sequencing runs/lanes from the same sample) should be summed up into a single column.

`geneCountData`
: A matrix of gene-level count data of non-negative integer values. The rows correspond to counts for each gene, the columns correspond to samples. Note that biological replicates should each get their own column, while the counts of technical replicates (i.e., several sequencing runs/lanes from the same sample) should be summed up into a single column. Must have the same dimensions as countData.

`design`
: A data frame consisting of all factors to be included in the analysis. All columns should be factors. Each column should represent a different variable, each row should represent a different sample. The number of rows must equal the number of columns in geneCountData and countData.

`geneIDs`
: A character vector of gene indentifiers for each splice junction. The length must equal the number of rows in countData.

`countbinIDs`
: A character vector of splice-junction-locus indentifiers for each splice junction. The length must equal the number of rows in countData.

`featureIntervals`
: Optional. A data.frame with 4 columns: "chr", "start", "end", and "strand". chr and strand should be character vectors or factors, start and end must be integers.

`transcripts`
: Optional. Character vector listing the transcripts that each splice junction belongs to. Some junctions may belong to more than one transcripts. In this case, transcripts should be separated with the "+" character.

Value

A JunctionSeqCountSet object. Additional data can be added to the

The constructor function above SHOULD NOT BE USED in normal operation. Instead you should use the `readJunctionSeqCounts` function.

See Also

`readJunctionSeqCounts`

---

plotDispEsts *Plot Fitted and Test-wise Dispersion*

Description

Plots the countbin-specific estimated dispersion and the fitted dispersion curve.

Usage

```
plotDispEsts( jscs, ymin,
              linecol="#ff000080",
              xlab = "mean of normalized counts",
              ylab = "dispersion",
              log = "xy", cex = 0.45, ... )
```

Arguments

jscs	A JunctionSeqCountSet. Usually created by runJunctionSeqAnalyses. Alternatively, this can be created manually by readJunctionSeqCounts. Dispersions and size factors must then be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions. Hypothesis tests must be performed by testJunctionsForDJU.
ymin	The minimum for the y-axis.
linecol	The line color to use for the fit line.
xlab	The label for the x-axis.
ylab	The label for the y-axis.
log	Logical value. Determines which axes to log-scale.
cex	The cex value to be passed to plot.
...	Additional options to pass to plotting functions, particularly graphical parameters.

---

plotJunctionSeqResultsForGene

Plot JunctionSeq results

Description

Creates one results plot for one gene.

Usage

```
plotJunctionSeqResultsForGene(geneID, jscs, flat.gff.data,
                              colorRed.FDR.threshold=0.05,
                              plot.type = "expr",
                              displayTranscripts = FALSE,
                              color = NULL,
                              use.vst = FALSE, use.log = FALSE,truncateBelowOne
                              exon.rescale.factor = 0.3,
                              label.p.vals = TRUE,
                              plot.lwd = 3,axes.lwd = plot.lwd, anno.lwd = plot.
                              par.cex = 1, anno.cex.text = 1,
                              anno.cex.axis=anno.cex.text, anno.cex.main = anno.
                              fit.countbin.names = TRUE,
                              debug.mode = FALSE,
                              plot.exon.results = FALSE, plot.splice.results = T
                              plot.untestable.results = FALSE,
                              show.strand.arrows = 20,
                              sort.features = TRUE,
                              drawCoordinates = TRUE,
                              arrows.length = 0.25,
                              title.main=NULL, title.ylab=NULL,
                              verbose=TRUE,
                              ...)
```

Arguments

geneID	Character string. The gene to the plotted.
jscs	A JunctionSeqCountSet. Usually created by runJunctionSeqAnalyses. Alternatively, this can be created manually by readJunctionSeqCounts. However in this case a number of additional steps will be necessary: Dispersions and size factors must then be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions. Hypothesis tests must be performed by testJunctionsForDJU. Effect sizes and parameter estimates must be created via estimateEffectSizes.
flat.gff.data	A data frame containing the annotation data. This can be extracted from a flat gff file using the readAnnotationData function.
colorRed.FDR.threshold	The adjusted-p-value threshold used to determine whether a feature should be marked as "significant" and colored pink. By default this will be the same as the FDR.threshold.
plot.type	Character string. Determines which plot to produce. Options are: "expr" for "expression", or mean normalized read counts by experimental condition, "rExpr" for "relative" expression relative to gene-level expression, "normCounts" for normalized read counts for each sample, and "rawCounts" for raw read counts for each sample.
displayTranscripts	Logical. If true, then the full set of annotated transcripts will be displayed below the expression plot (to a maximum of 42 different TX).
color	A vector of R colors, named for each possible value of condition. By default, it will attempt to choose reasonable colors for each condition.
use.vst	Logical. If TRUE, all plots will be scaled via a variance stabilizing transform.
use.log	Logical. If TRUE, all plots will be log-scaled.
truncateBelowOne	Logical. If TRUE, all values between 0 and 1 will be drawn with a linear scale. If use.log is FALSE, this does nothing.
exon.rescale.factor	Numeric. Exons will be proportionately scaled-up so that the exonic regions make up this fraction of the horizontal plotting area. If negative, exons and introns will be plotted to a common scale.
label.p.vals	Logical. If TRUE, then statistically significant p-values will be labelled.
plot.lwd	the line width for the plotting lines.
axes.lwd	the line width for the axes.
anno.lwd	the line width for the various other annotation lines.
par.cex	The base cex value to be passed to par() immediately before all plots are created. See par.
anno.cex.text	The font size multiplier for most annotation text. This will be multiplied by a factor of the par.cex value. More specifically: The cex value to be passed to all function calls that take graphical parameters. See par.
anno.cex.axis	The font size multiplier for the axis text. This will be multiplied by a factor of the par.cex value. More specifically: The cex.axis value to be passed to all function calls that take graphical parameters. See par.

`anno.cex.main`
: The font size multiplier for the main title text. This will be multiplied by a factor of the par.cex value. More specifically: The cex.main value to be passed to all function calls that take graphical parameters. See `par`.

`fit.countbin.names`
: Logical. If `TRUE`, then splice-junction-locus labels should be rescaled to fit in whatever horizontal space is available.

`debug.mode`
: Logical. If `TRUE`, print additional debugging information during execution.

`plot.exon.results`
: Logical. If TRUE, plot results for exons. Technically speaking, JunctionSeq can be used to do DEXSeq-style analyses on exon partitions. However this functionality is for advanced users only.

`plot.splice.results`
: Logical. If TRUE, plot results for splice junctions. For advanced users only.

`plot.novel.splice.results`
: Logical. If TRUE, plot results for novel splice junctions. If false, novel splice junctions will be ignored.

`plot.untestable.results`
: Logical. If TRUE, plots splice junctions that had coverage that was too low to be tested.

`show.strand.arrows`
: The number of strand-direction arrows to display.

`sort.features`
: Logical. If `TRUE`, sort features by genomic position.

`drawCoordinates`
: Whether to label the genomic coordinates at the bottom of the plot.

`arrows.length`
: The length of the strand-direction arrows, in inches.

`title.main`
: Character string. Overrides the default main plot title.

`title.ylab`
: Character string. Overrides the default y-axis label.

`verbose`
: if TRUE, send debugging and progress messages to the console / stdout.

`...`
: Additional options to pass to plotting functions, particularly graphical parameters.

`plotMA` *Generate a MA-Plot*

Description

TODO

Usage

```
plotMA(jscs,
       FDR.threshold = 0.05,
       fc.name = NULL,
       fc.thresh = 1,
       use.pch = 19,
```

```
        smooth.nbin = 256,
        ylim = c( 1 / 1000,1000),
        use.smoothScatter = TRUE,
        label.counts = TRUE,
        label.axes = c(TRUE,TRUE,FALSE,FALSE),
        show.labels = TRUE,
        par.cex = 1, points.cex = 1, text.cex = 1,
        lines.cex = 8,
        anno.lwd = 2,
        miniTicks = TRUE, ...)
```

Arguments

jscs	A JunctionSeqCountSet. Usually created by runJunctionSeqAnalyses. Alternatively, this can be created manually by readJunctionSeqCounts. However in this case a number of additional steps will be necessary: Dispersions and size factors must then be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions. Hypothesis tests must be performed by testJunctionsForDJU. Effect sizes and parameter estimates must be created via estimateEffectSizes.
FDR.threshold	The FDR threshold used to color dots. Tests with an adjusted-p-value more significant than this threshold will be marked in red.
fc.name	The name of the column to take from fData(jscs).
fc.thresh	The fold-change threshold required to count a significant locus in the count labels. It will also draw horizontal lines at this threshold.
use.pch	The value of pch to pass to the points call.
smooth.nbin	The number of bins to smooth, for the density plot, if use.smoothScatter is TRUE.
use.smoothScatter	Logical. If TRUE, non-significant genes will be ploted with density shading.
ylim	The y-axis limits.
label.counts	Logical. If TRUE, include labels showing the number of loci that pass both the statistical-significance and fold-change threshold in each direction.
label.axes	Logical vector. Whether to label each axis. Must have length 4; each corresponds to the bottom, left, top, and right axes respectively.
show.labels	Logical. If TRUE, include all titles and axes labels.
par.cex	The cex value to be passed to par.
points.cex	The cex value to be passed to points.
text.cex	The cex value to be passed to text.
lines.cex	The cex value to be passed to lines, box, and similar.
anno.lwd	The lwd value to be passed to lines, box, axis, and similar.
miniTicks	Logical. If TRUE, then include "mini tick marks" on the x and y axes.
...	Additional graphical parameters.

readAnnotationData *Read junctionSeq annotation files produced by QoRTs.*

Description

This function reads the "flattened" gff annotation file created by QoRTs. This annotation file contains all the gene, transcript, exon, and junction ID's and their loci.

Usage

```
readAnnotationData(flat.gff.file)
```

Arguments

`flat.gff.file`
: Character string. The filename of the "flat" gff annotation file. The file may be gzip-compressed. This "flat" gff file must be produced by the QoRTs jar utility using the makeFlatGtf or mergeNovelSplices functions (depending on whether inclusion of novel splice junctions is desired).

readJunctionSeqCounts *Read junctionSeq count files*

Description

This function reads count data (usually produced by QoRTs) and compiles them into a `JunctionSeqCountSet`.

For most purposes this function should not be needed, but might be useful to advanced users performing non-standard analyses. This and all other standard analysis functions can be run automatically via the `runJunctionSeqAnalyses` function.

Usage

```
readJunctionSeqCounts(countfiles,
                      countdata,
                      samplenames,
                      design,
                      flat.gff.file,
                      use.splice.sites = TRUE,
                      use.novel.splice.sites = TRUE,
                      use.exons = FALSE,
                      verbose = TRUE)
```

Arguments

countfiles	Character vector. The filenames of the count files generated by QoRTs. The counts must all be generated using equivalent QoRTs parameters. The strandedness must be the same, as well as the inclusion of novel junctions.
countdata	List. An alternative parameterization. Instead of supplying count files using the countfiles parameter, you can pass a list of data frames, one for each sample. Each data frame should contain two columns: the first should be the feature id and the second should be the counts. This list must have the same length as the samplenames parameter.
samplenames	Character vector. A vector of full sample names, in the same order as the countfiles parameter.
design	A data frame containing the condition variable and all desired covariates. All variables should be factors.
flat.gff.file	Character string. The filename of the "flat" gff annotation file. Can be gzip-compressed. This "flat" gff file must be produced by the QoRTs jar utility using the makeFlatGtf or mergeNovelSplices functions (depending on whether inclusion of novel splice junctions is desired). NOTE: This option is technically optional, but strongly recommended. If it is not included, then attempts to plot the results will crash unless (non-default) options are used to deactivate the plotting of genomic coordinates and transcript information
use.splice.sites	FOR ADVANCED USERS ONLY: Logical value. If TRUE, splice junctions will be included in the analyses.
use.novel.splice.sites	Logical value. If TRUE, splice junctions that are marked as novel in the gff annotation will be included in the analysis. Note that this is does nothing if there are no novel splice junctions in the count tables.
use.exons	FOR ADVANCED USERS ONLY: Logical value. If TRUE, exon segments will be included in the analyses.
verbose	if TRUE, send debugging and progress messages to the console / stdout.

Value

A JunctionSeqCountSet.

---

runJunctionSeqAnalyses

Run a JunctionSeq analysis.

Description

This function runs a complete analysis from start to finish. It internally calls functions readAnnotationData, readJunctionSeqCounts, estimateSizeFactors, estimateJunctionSeqDispersions, fitDispersionFunction, testJunctionsForDJU, and estimateEffectSizes.

Usage

```
runJunctionSeqAnalyses(sample.files, sample.names, condition,
                       flat.gff.file,
                       outfile.prefix, saveState = FALSE,
                       use.splice.sites = TRUE,
                       use.novel.splice.sites = TRUE,
                       use.exons = FALSE,
                       nCores = 1,
                       use.covars = NULL,
                       test.formula0 = formula(~ sample + countbin),
                       test.formula1 = formula(~ sample + countbin + condition
                       effect.formula = formula(~ condition + countbin + condi
                       gzip.output = TRUE,
                       test.aggregated.genes = FALSE,
                       use.alternate.method = TRUE,
                       verbose = TRUE)
```

Arguments

`sample.files` A character vector of sample files. Each sample file is a simple tab-delimited file containing two columns. The first column contains the feature name, using the format GENE_ID:SPLICE_SITE_ID, where GENE_ID can be any alphanumeric string, and SPLICE_SITE_ID is a 3-digit number. The 2nd column is the read count for that feature, as a non-negative integer. Do not use normalized read counts, RPKM, FPKM, or anything other than raw read counts, as this will conflict with the DEXSeq normalization.

`sample.names` A character vector of sample names. This must have the same length as sample.files, and should be in the same order.

`condition` A factor vector of condition values. This must have the same length as sample.files and sample.names, and should be listed in the same order.

`flat.gff.file` A flattened gff-formatted annotation file from which the gene counts were generated. Technically optional, but STRONGLY RECOMMENDED, as the annotation data WILL be required by plotting functions.

`outfile.prefix` The prefix of the output files to be written. By default no output files will be created.

`saveState` If TRUE and if outfile.prefix is non-null, then the ecs and res objects will be saved to disk as RData files after all analysis is complete.

`use.splice.sites` FOR ADVANCED USERS ONLY: Logical value. If `TRUE`, splice junctions will be included in the analyses.

`use.novel.splice.sites` Logical value. If `TRUE`, splice junctions that are marked as novel in the gff annotation will be included in the analysis. Note that this is does nothing if there are no novel splice junctions in the count tables.

`use.exons` FOR ADVANCED USERS ONLY: Logical value. If `TRUE`, exon segments will be included in the analyses.

`nCores` The number of cores to use. Note that multicore functionality may not be available on all platforms. JunctionSeq attempts to use the BiocParallell package if it can be found installed. Otherwise it will attempt to fallback to the multicore package. If neither package can be found it will fallback to single core execution.

`use.covars` Optional: for advanced users. A data frame containing covariate factors. The names must be included in the model formulas.

`test.formula0`
For advanced users. The base formula for the null hypothesis model used in the hypothesis tests.
NOTE: the biological condition to be tested must be named "condition".

`test.formula1`
For advanced users. The base formula for the alternate hypothesis model used in the hypothesis tests.
NOTE: the biological condition to be tested must be named "condition".

`effect.formula`
For advanced users. The base formula for the model used for effect size estimation.
NOTE: the biological condition to be tested must be named "condition".

`gzip.output` Determines whether the text output should be gzip compressed or in plaintext.

`test.aggregated.genes`
Logical value. Whether to attempt to test "aggregate genes" which consist of multiple genes that overlap with one another. Note that inclusion of aggregate genes may affect the false discovery rate, since by their very nature aggregate genes will often show differential splice junction usage, as the two genes will often be regulated independently.

`use.alternate.method`
DEPRECIATED: whether to use the recommended JunctionSeq model framework, or to fallback to the DEXSeq-style model framework.

`verbose` if TRUE, send debugging and progress messages to the console / stdout.

`testJunctionsForDJU`
Test Junctions for Differential Junction Usage

Description

This function runs the hypothesis tests for differential junction usage.

For most purposes this function should not be needed, but might be useful to advanced users performing non-standard analyses. This and all other standard analysis functions can be run automatically via the `runJunctionSeqAnalyses` function.

Usage

```
testJunctionsForDJU( jscs,
          test.formula0 = formula(~ sample + countbin),
          test.formula1 = formula(~ sample + countbin + condition : countbin),
          dispColumn="dispersion", nCores=1,
          use.alternate.method = TRUE,
          verbose = TRUE)
```

Arguments

jscs	A JunctionSeqCountSet. Usually initially created by readJunctionSeqCounts. Dispersions and size factors must be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions.
test.formula0	The formula for the null hypothesis. Note that the condition to be tested must be named "condition".
test.formula1	The formula for the alternative hypothesis. Note that the condition to be tested must be named "condition".
dispColumn	Character value. The name of the fData(jscs) column in which the model dispersion is stored.
nCores	The number of cores to use. Note that multicore functionality may not be available on all platforms. JunctionSeq attempts to use the BiocParallell package if it can be found installed. Otherwise it will attempt to fallback to the multicore package. If neither package can be found it will fallback to single core execution.
use.alternate.method	ADVANCED USERS ONLY: Logical value. Determines whether to use the JunctionSeq standard model framework, or the DEXSeq model framework.
verbose	if TRUE, send debugging and progress messages to the console / stdout.

Value

A JunctionSeqCountSet, with hypothesis test results included.

---

writeBedTrack *Write splice junction browser tracks*

Description

This function saves the JunctionSeq results in the form of a set of "bed" files designed for use with the UCSC genome browser.

Usage

```
writeExprBedTrack(file, jscs,
                  trackLine,
                  only.with.sig.gene = TRUE,
                  only.sig = FALSE,
                  only.testable = TRUE,
                  group.RGB,
                  use.score = FALSE,
                  FDR.threshold = 0.05,
                  count.digits = 1,
                  includeGeneID = FALSE,
                  includeLocusID = TRUE,
                  includeGroupID = TRUE)
```

```
writeSigBedTrack(file,
                 jscs,
                 trackLine,
                 only.sig = TRUE,
                 only.testable = TRUE,
                 sig.RGB = "255,0,0",
                 nonsig.RGB = "0,0,0",
                 use.score = TRUE,
                 FDR.threshold = 0.05,
                 pval.digits = 4,
                 includeGeneID = FALSE,
                 includeLocusID = TRUE)
```

Arguments

file	Character string. File path for the output bed file.
jscs	A JunctionSeqCountSet. Usually created by runJunctionSeqAnalyses. Alternatively, this can be created manually by readJunctionSeqCounts. However in this case a number of additional steps will be necessary: Dispersions and size factors must then be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions. Hypothesis tests must be performed by testJunctionsForDJU. Effect sizes and parameter estimates must be created via estimateEffectSizes.
trackLine	The "track line" of the bed file. In other words, the first line of the file. By default JunctionSeq will attempt to automatically generate a reasonable track line.
only.with.sig.gene	Logical. If TRUE, only genes containing statistically significant results will be included.
only.sig	Logical. If TRUE, only statistically significant splice junctions will be included.
only.testable	Logical. If TRUE, only junction loci with sufficiently high expression to be tested will be included.
sig.RGB	Character string. The RGB color for significant genes. Must be in the format "r,g,b", with each value ranging from 0 to 255.
nonsig.RGB	Character string. The RGB color for non-significant junctions junctions. Must be in the format "r,g,b", with each value ranging from 0 to 255.
group.RGB	Character string. The RGB color used for each experimental group. Must be in the format "r,g,b", with each value ranging from 0 to 255. Must have a length equal to the number of experimental condition values.
use.score	Logical. If TRUE, score each junction based on the p-value.
FDR.threshold	Numeric. The FDR-adjusted p-value threshold to use to assign statistical significance.
count.digits	Numeric. The number of digits after the decimal point to include for the mean normalized counts.
pval.digits	Numeric. The number of digits after the decimal point to include for the p-values.

includeGeneID
: Logical. If TRUE, include the ID of the gene in the "name" field of each line.

includeLocusID
: Logical. If TRUE, include the ID of the junction locus in the "name" field of each line.

includeGroupID
: Logical. If TRUE, include the ID of the group in the "name" field of each line.

---

writeCompleteResults
: *Produce output data files, given annotation files and DEXSeq exonCountSet object and DEXSeq results data.*

Description

This function takes the raw DEXSeq results and merges in feature annotations, as well as calculating and merging in a number of different normalized and fitted values for each level of the condition variable.

Usage

```
writeCompleteResults(jscs, outfile.prefix,
                         gzip.output = TRUE,
                         FDR.threshold = 0.05,
                         save.allGenes = TRUE, save.sigGenes = TRUE,
                         save.fit = TRUE, save.VST = TRUE,
                         save.bedTracks = TRUE,
                         save.jscs = FALSE,
                         verbose = TRUE)
```

Arguments

jscs
: A JunctionSeqCountSet. Usually created by runJunctionSeqAnalyses. Alternatively, this can be created manually by readJunctionSeqCounts. However in this case a number of additional steps will be necessary: Dispersions and size factors must then be set, usually using functions estimateSizeFactors and estimateJunctionSeqDispersions. Hypothesis tests must be performed by testJunctionsForDJU. Effect sizes and parameter estimates must be created via estimateEffectSizes.

outfile.prefix
: A string indicating the filename prefix where output files should be saved.

gzip.output
: Logical. If TRUE, then all ".txt" text files should be gzip-compressed to save space.

FDR.threshold
: The adjusted-p-value threshold used to determine statistical significance.

save.allGenes
: Logical. Whether to save files containing data for all genes.

save.sigGenes
: Logical. Whether to save a separate set of files containing data for only the significant genes. If this and save.allGenes are both true then two sets of files will be generated.

save.fit	Logical. Whether to save model fit data.
save.VST	Logical. Whether to save VST-transformed data.
save.bedTracks	Logical. Whether to save "bed" junction coverage tracks.
save.jscs	Logical. Whether to the entire JunctionSeqCountSet object. Default is FALSE.
verbose	A boolean flag indicating whether or not to print progress information during execution. (Default=FALSE)

Details

Saves a wide variety of data from the analyses.

Index

axis, *18*

box, *18*
buildAllPlots, 2
buildAllPlotsForGene, 5

estimateEffectSizes, *2*, *6*, 8, *16*, *18*, *20*, *24, 25*
estimateJunctionSeqDispersions, *2*, *6*, *8*, 9, *9–12*, *15, 16*, *18*, *20*, *23–25*
estimateSizeFactors, *2*, *6*, *8, 9*, 10, *10–12*, *15, 16*, *18*, *20*, *23–25*

fitDispersionFunction, 10, *20*

generateAllExpressionEstimates, 11
generateCompleteResults, *11*, 12
graphical parameters, *4*, *7*, *16, 17*

JunctionSeqCountSet *(JunctionSeqCountSet-class)*, 13
JunctionSeqCountSet-class, 13

lines, *18*

newJunctionSeqCountSet *(JunctionSeqCountSet-class)*, 13

par, *4*, *7*, *16–18*
plotDispEsts, 14
plotJunctionSeqResultsForGene, 15
plotMA, 17
points, *18*

readAnnotationData, *3*, *6*, *16*, 19, *20*
readJunctionSeqCounts, *2*, *6*, *8–12*, *14–16*, *18*, 19, *20*, *23–25*
runJunctionSeqAnalyses, *2*, *6*, *8–12*, *15, 16*, *18, 19*, 20, *22*, *24, 25*

testJunctionsForDJU, *2*, *6*, *11, 12*, *15, 16*, *18*, *20*, 22, *24, 25*

text, *18*

writeBedTrack, 23
writeCompleteResults, 25
writeExprBedTrack *(writeBedTrack)*, 23
writeSigBedTrack *(writeBedTrack)*, 23