Package 'JunctionSeq'

June 11, 2015

Version 0.3.41

Date 2015-06-11

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,locfit

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` *Create and save a full battery of JunctionSeq expression plots.*

Description

Saves a large battery of plots displaying the analysis results, for the purposes of data visualization. By default it saves a full set of plots for every gene that shows statistical significance and the adjusted-p < 0.01 level. Alternatively, it can be supplied with a specific gene list using the `gene.list` parameter, and will plot those specific genes.

Note that this function has MANY parameters, allowing the user to tweak the appearance of the plots to suit their particular needs and preferences. Don't be daunted: the default parameters are probably fine for most purposes.

Usage

```
buildAllPlots(jscs,
              outfile.prefix = "./",
              gene.list = NULL, FDR.threshold = 0.01,
              use.plotting.device = "png",
              use.vst=FALSE,use.log = TRUE, truncateBelowOne = TRUE,
              exon.rescale.factor = 0.3,
              subdirectories.by.type = TRUE,
              ma.plot=FALSE, variance.plot=FALSE,
              with.TX=TRUE,without.TX=TRUE,
              expr.plot=TRUE,normCounts.plot=TRUE,
              rExpr.plot=FALSE,rawCounts.plot=FALSE,
              colorRed.FDR.threshold = FDR.threshold,
              color=NULL,
              plot.gene.level.expression = NULL,
              plot.exon.results, plot.junction.results, plot.novel.junction.re
              plot.untestable.results = FALSE,
              plot.lwd=3, axes.lwd = plot.lwd, anno.lwd = plot.lwd,
              par.cex = 1, anno.cex.text = 1, anno.cex.axis = anno.cex.text, a
              drawCoordinates = TRUE,
              yAxisLabels.inExponentialForm = FALSE,
              show.strand.arrows = 10, arrows.length = 0.125,
              graph.margins = c(2,3,3,2),
              base.plot.height = 22.222, base.plot.width = 22.222,
              base.plot.units = "in",
              GENE.annotation.relative.height = 0.2,
              TX.annotation.relative.height = 0.025,
              autoscale.height.to.fit.TX.annotation = TRUE,
              autoscale.width.to.fit.bins = TRUE,
              plotting.device.params = list(),
              number.plots = TRUE,
              condition.legend.text, include.TX.names = TRUE,
              draw.start.end.sites = 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 testForDiffUsage. Effect sizes and parameter estimates must be created via estimateEffectSizes.
outfile.prefix	The prefix file path to save the images to.
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.
subdirectories.by.type	Logical value. If TRUE, then subdirectories will be created in the outfile.prefix directory, containing each plot type.
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.gene.level.expression
: Logical value. If TRUE, gene-level expression (when applicable) will be plotted beside the sub-element-specific expression in a small seperate plotting box. The gene-level expression will NOT be plotted for the "relative expression" plots (since you can't meaningfully plot gene-level expression relative to itself).

plot.exon.results
: Logical. If TRUE, plot results for exons. By default everything that was tested will be plotted.

plot.junction.results
: Logical. If TRUE, plot results for splice junctions. By default everything that was tested will be plotted.

plot.novel.junction.results
: Logical. If TRUE, plot results for novel splice junctions. If false, novel splice junctions will be ignored. By default everything that was tested will be plotted.

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.

yAxisLabels.inExponentialForm
: Logical. If TRUE, then the y-axis will be labelled in exponential form.

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

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

`graph.margins`
: Numeric vector of length 4. These margins values used (as if for par("mar")) for the main graph. The lower part of the plot uses the same left and right margins.

`base.plot.height`
: The base height of the standard-sized plots. Plots that include the full transcript annotation will be expanded by the height of these additional rows. See the withTxPlot.height.multiplier parameter, below.

`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.

`GENE.annotation.relative.height`
: The height of the "gene track" displayed underneath the main graph, relative to the height of the main graph. By default it is 20 percent.

`TX.annotation.relative.height`
: For all plots that draw the annotated-transcript set (when the `with.TX` parameter is `TRUE`), this sets the height of each transcript, as a fraction of the height of the main graph. By default it is 2.5 percent.

`autoscale.height.to.fit.TX.annotation`
: Plots that include the full transcript annotation generally need to have a larger height in order to maintain readability. By default, all plots that include transcripts will be expanded vertically by the height of the additional transcripts. This maintains the same appearance and aspect ratio of the main graph, but also means that the height of the plot will differ between genes. This parameter can be used to override that behavior if a specific figure size is desired. If `TRUE`, the `base.plot.height` will be used as the height of the plot, regardless of how many transcripts are included.

`autoscale.width.to.fit.bins`
: Integer value. JunctionSeq will automatically go to great lengths to autofit the data in a readable way. By default, any plots that have more than 35 plotting columns will be widened linearly to fit the excess columns. This parameter can be used to change that value, or turn it off entirely by setting this parameter to `NA`.

`condition.legend.text`
: List or named vector of character strings. This optional parameter can be used to assign labels to each condition variable values. It should be a list or named vector with length equal to `factor(condition)`. Each element should be named with one of the values from `factor(condition)`, and should contain the label. They will be listed in this order in the figure legend.

`include.TX.names`
: Logical value. If `TRUE`, then for the plots that include the annotated transcript, the transcript names will be listed. The labels will be drawn at half the size of `anno.cex.text`.

`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.

`draw.start.end.sites`
: Logical value. If `TRUE`, then transcript start/end sites will be marked on the main gene annotation.

`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` *Create and save one or more JunctionSeq expression plots.*

Description

Generates and saves one or more plots, displaying counts or averages for all counting bins across one particular gene. The parameters `expr.plot`, `normCounts.plot`, `rExpr.plot`, and `rawCounts.plot` determine which plot types are to be generated, and the parameters `with.TX` and `without.TX` determines whether these plots should include or not include the full transcript information, or if separate plots should be generated with and without the full transcript information.

Note that this function has MANY parameters, allowing the user to tweak the behavior and appearance of the plots to suit their particular needs and preferences. Don't be daunted: the default parameters are probably fine for most purposes.

Usage

```
buildAllPlotsForGene(geneID, jscs,
                     outfile.prefix = "./",
                     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=FALSE,rawCounts.plot=FALSE,
                     colorRed.FDR.threshold = 0.01,
                     color=NULL,
                     plot.gene.level.expression = NULL,
                     plot.exon.results, plot.junction.results, plot.novel.juncti
                     plot.untestable.results = FALSE,
                     plot.lwd=3, axes.lwd = plot.lwd, anno.lwd = plot.lwd,
                     par.cex = 1, anno.cex.text = 1,
                     anno.cex.axis = anno.cex.text, anno.cex.main = anno.cex.tex
                     drawCoordinates = TRUE,
                     yAxisLabels.inExponentialForm = FALSE,
                     show.strand.arrows = 10, arrows.length = 0.125,
```

```
                    graph.margins = c(2,3,3,2),
                    base.plot.height = 22.222, base.plot.width = 22.222,
                    base.plot.units = "in",
                    GENE.annotation.relative.height = 0.2, TX.annotation.relati
                    autoscale.height.to.fit.TX.annotation = TRUE,
                    autoscale.width.to.fit.bins = 35,
                    plotting.device.params = list(),
                    condition.legend.text, include.TX.names = TRUE,
                    draw.start.end.sites = TRUE,
                    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 testForDiffUsage. Effect sizes and parameter estimates must be created via estimateEffectSizes.
outfile.prefix	Character string or vector. Sets the prefix file path where image files should be saved. Optionally it can be a vector of strings, assigning a different file prefix to each plot.
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.gene.level.expression
: Logical value. If TRUE, gene-level expression (when applicable) will be plotted beside the sub-element-specific expression in a small seperate plotting box. The gene-level expression will NOT be plotted for the "relative expression" plots (since you can't meaningfully plot gene-level expression relative to itself).

plot.exon.results
: Logical. If TRUE, plot results for exons. By default everything that was tested will be plotted.

plot.junction.results
: Logical. If TRUE, plot results for splice junctions. By default everything that was tested will be plotted.

plot.novel.junction.results
: Logical. If TRUE, plot results for novel splice junctions. If false, novel splice junctions will be ignored. By default everything that was tested will be plotted.

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.

yAxisLabels.inExponentialForm
: Logical. If TRUE, then the y-axis will be labelled in exponential form.

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

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

`graph.margins`
: Numeric vector of length 4. These margins values used (as if for par("mar")) for the main graph. The lower part of the plot uses the same left and right margins.

`base.plot.height`
: The base height of the standard-sized plots. Plots that include the full transcript annotation will be expanded by the height of these additional rows. See the withTxPlot.height.multiplier parameter, below.

`base.plot.width`
: The base width of the plots (plots with a large number of features may be scaled up, see parameter `autoscale.width.to.fit.bins`).

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

`GENE.annotation.relative.height`
: The height of the "gene track" displayed underneath the main graph, relative to the height of the main graph. By default it is 20 percent.

`TX.annotation.relative.height`
: For all plots that draw the annotated-transcript set (when the `with.TX` parameter is `TRUE`), this sets the height of each transcript, as a fraction of the height of the main graph. By default it is 2.5 percent.

`autoscale.height.to.fit.TX.annotation`
: Plots that include the full transcript annotation generally need to have a larger height in order to maintain readability. By default, all plots that include transcripts will be expanded vertically by the height of the additional transcripts. This maintains the same appearance and aspect ratio of the main graph, but also means that the height of the plot will differ between genes. This parameter can be used to override that behavior if a specific figure size is desired. If `TRUE`, the `base.plot.height` will be used as the height of the plot, regardless of how many transcripts are included.

`autoscale.width.to.fit.bins`
: Integer value. JunctionSeq will automatically go to great lengths to autofit the data in a readable way. By default, any plots that have more than 35 plotting columns will be widened linearly to fit the excess columns. This parameter can be used to change that value, or turn it off entirely by setting this parameter to `NA`.

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

`condition.legend.text`
: List or named vector of character strings. This optional parameter can be used to assign labels to each condition variable values. It should be a list or named vector with length equal to `factor(condition)`. Each element should be named with one of the values from `factor(condition)`, and should contain the label. They will be listed in this order in the figure legend.

`include.TX.names`
: Logical value. If `TRUE`, then for the plots that include the annotated transcript, the transcript names will be listed. The labels will be drawn at half the size of `anno.cex.text`.

`draw.start.end.sites`
: Logical value. If `TRUE`, then transcript start/end sites will be marked on the main gene annotation.

`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 Sizes, parameter estimates, etc.

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 each analysis.

This function is called internally by the `runJunctionSeqAnalyses` function, and thus for most purposes users should not need to call this function directly. It may be useful to advanced users performing non-standard analyses.

Usage

```
estimateEffectSizes(jscs,
     effect.formula = formula(~ condition + countbin + condition : countbin),
     geneLevel.formula = formula(~ condition),
     calculate.geneLevel.expression = TRUE,
     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".

`geneLevel.formula`
: For advanced users. The base formula for the model used to estimate total gene-level expression.

  NOTE: the biological condition to be tested must be named "condition".

`calculate.geneLevel.expression`
: Logical value. If `TRUE`, gene-level expression will be estimated using the same maximum-likelihood method used in other analyses. Default: `TRUE`.

`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).

This function is called internally by the `runJunctionSeqAnalyses` function, and thus for most purposes users should not need to call this function directly. It may be useful to advanced users performing non-standard analyses.

Usage

```
estimateJunctionSeqDispersions(jscs,
        test.formula1 = formula(~ sample + countbin + condition : countbin),
        meanCountTestableThreshold=7.5, 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".

`meanCountTestableThreshold`
: Numeric value. Features with a total mean normalized count of less than this value will be excluded from the analyses.

`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 which model framework to use.

`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.

This function is called internally by the `runJunctionSeqAnalyses` function, and thus for most purposes users should not need to call this function directly. It may be useful to advanced users performing non-standard analyses.

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.

This function is called internally by the `runJunctionSeqAnalyses` function, and thus for most purposes users should not need to call this function directly. It may be useful to advanced users performing non-standard analyses.

Usage

```
fitDispersionFunction(jscs, fitType = c("parametric","local","mean"),
                      fitDispersionsForExonsAndJunctionsSeparately = TRUE,
                      advancedMode = TRUE,
                      verbose = TRUE);
```

Arguments

`jscs`	A `JunctionSeqCountSet`. Usually initially created by `readJunctionSeqCounts`. Size factors must be set, usually using functions `estimateSizeFactors` and `estimateJunctionSeqDispersions`.
`fitType`	Which dispersion fit method to use. Not that currently only the parametric (default) method is supported. The other methods are currently nonfunctional and will be added in future releases.
`fitDispersionsForExonsAndJunctionsSeparately`	When running a "hybrid" analysis in which both exons and splice junctions are being tested simultaniously, this parameter determines whether a single fitted dispersion model should be fitted for both exons and splice junctions, or if separate fitted dispersions should be calculated for each. By default the dispersions are run separately.
`advancedMode`	Switches between the current default method and an older variant.
`verbose`	if TRUE, send debugging and progress messages to the console / stdout.

Value

A JunctionSeqCountSet, with dispersion results included.

`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, ylim, xlim, linecol = c("#ff000080","#ff990080"),
             xlab = "mean of normalized counts",
             ylab = "dispersion",
             miniTicks = TRUE,
             cex = 0.45, pch = c(1,4),
             use.smoothScatter = FALSE, smooth.nbin = 512, nrpoints = 100,
             plot.exon.results = TRUE, plot.junction.results = 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 testForDiffUsage.
ylim	The plotting range for the y-axis.
xlim	The plotting range for the x-axis.
linecol	Character vector of length 2. The line color to use for the fit line. If the fits were performed separately for exons and junctions, the junction line will be drawn with the second color (orange, by default).
xlab	The label for the x-axis.
ylab	The label for the y-axis.
miniTicks	Whether or not to plot smaller ticks at the tenth-decades.
cex	The cex value to be passed to plot, determining the size of the plotted points.
pch	Numeric vector of length 2. Contains the pch parameters for the exon features and the junction features. This determines the character used in plotting. By default the exons are plotted as circles, and junctions are plotted as X's.
use.smoothScatter	Logical. If TRUE, features will be ploted with density shading rather than having each point plotted.
smooth.nbin	The number of bins to smooth, for the density plot, if use.smoothScatter is TRUE.
nrpoints	The number of extra points to plot, if use.smoothScatter is TRUE.
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.junction.results`
: Logical. If `TRUE`, plot results for splice junctions. For advanced users only.

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

`plotJunctionSeqResultsForGene` *Generate a JunctionSeq expression plot.*

Description

Creates one results plot for one gene. Note that this function does not call a plotting device, so it will simply plot to the "current" device. If you want to automatically save images to file, use `buildAllPlotsForGene`, which internally calls this function.

Note that this function has MANY parameters, allowing the user to tweak the appearance of the plots to suit their particular needs and preferences. Don't be daunted: the default parameters are probably fine for most purposes.

Usage

```
plotJunctionSeqResultsForGene(geneID, jscs,
                              colorRed.FDR.threshold=0.05,
                              plot.type = "expr",
                              displayTranscripts = FALSE,
                              color = NULL,
                              use.vst = FALSE, use.log = TRUE,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,
                              plot.gene.level.expression = NULL,
                              plot.exon.results, plot.junction.results, plot.nov
                              plot.untestable.results = FALSE, draw.untestable.a
                              show.strand.arrows = 10, arrows.length = 0.125,
                              sort.features = TRUE,
                              drawCoordinates = TRUE,
                              yAxisLabels.inExponentialForm = FALSE,
                              title.main=NULL, title.ylab=NULL,
                              graph.margins = c(2,3,3,2),
                              GENE.annotation.relative.height = 0.2,
                              TX.annotation.relative.height = 0.025,
                              condition.legend.text = NULL, include.TX.names = T
                              draw.start.end.sites = TRUE,
                              verbose=TRUE, debug.mode = FALSE,
                              ...)
```

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 testForDiffUsage. Effect sizes and parameter estimates must be created via estimateEffectSizes.
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.

`plot.gene.level.expression`
: Logical value. If `TRUE`, gene-level expression (when applicable) will be plotted beside the sub-element-specific expression in a small seperate plotting box. The gene-level expression will NOT be plotted for the "relative expression" plots (since you can't meaningfully plot gene-level expression relative to itself).

`plot.exon.results`
: Logical. If `TRUE`, plot results for exons. By default everything that was tested will be plotted.

`plot.junction.results`
: Logical. If `TRUE`, plot results for splice junctions. By default everything that was tested will be plotted.

`plot.novel.junction.results`
: Logical. If `TRUE`, plot results for novel splice junctions. If false, novel splice junctions will be ignored. By default everything that was tested will be plotted.

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

`draw.untestable.annotation`
: Logical. If TRUE, draws the annotation for splice junctions that had coverage that was too low to be tested.

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

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

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

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

`yAxisLabels.inExponentialForm`
: Logical. If `TRUE`, then the y-axis will be labelled in exponential form.

`graph.margins`
: Numeric vector of length 4. These margins values used (as if for par("mar")) for the main graph. The lower part of the plot uses the same left and right margins.

`GENE.annotation.relative.height`
: The height of the "gene track" displayed underneath the main graph, relative to the height of the main graph. By default it is 20 percent.

`TX.annotation.relative.height`
: For all plots that draw the annotated-transcript set (when the `with.TX` parameter is `TRUE`), this sets the height of each transcript, as a fraction of the height of the main graph. By default it is 2.5 percent.

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

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

condition.legend.text
: List or named vector of character strings. This optional parameter can be used to assign labels to each condition variable values. It should be a list or named vector with length equal to `factor(condition)`. Each element should be named with one of the values from `factor(condition)`, and should contain the label. They will be listed in this order in the figure legend.

include.TX.names
: Logical value. If `TRUE`, then for the plots that include the annotated transcript, the transcript names will be listed. The labels will be drawn at half the size of `anno.cex.text`.

draw.start.end.sites
: Logical value. If `TRUE`, then transcript start/end sites will be marked on the main gene annotation.

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

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

...
: 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 `testForDiffUsage`. 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.

use.smoothScatter
: Logical. If TRUE, non-significant genes will be ploted with density shading.

smooth.nbin
: The number of bins to smooth, for the density plot, if use.smoothScatter is TRUE.

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 loads read-count data (usually produced by QoRTs) and compiles them into a `JunctionSeqCountSet` object.

This function is called internally by the `runJunctionSeqAnalyses` function, and thus for most purposes users should not need to call this function directly. It may be useful to advanced users performing non-standard analyses.

Usage

```
readJunctionSeqCounts(countfiles, countdata,
                      samplenames, design,
                      flat.gff.file,
                      analysis.type = c("junctionsAndExons","junctionsOnly","exo
                      nCores = 1,
                      use.exons, use.junctions,
                      use.novel.junctions = TRUE,
                      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
`analysis.type`	Character string. One of "junctionsAndExons", "junctionsOnly", or "exonsOnly". This parameter determines what type of analysis is to be performed. By default

JunctionSeq tests both splice junction loci and exonic regions for differential usage (a "hybrid" analysis). This parameter can be used to limit analyses specifically to either splice junction loci or exonic regions.

`nCores` The number of cores to use. Note that multicore functionality may not be available on all platforms.

`use.exons` Logical value. This is an alternate parameterization of the `analysis.type` parameter. If `TRUE`, then exonic region loci will be included in the analyses and will be tested for differential usage. If this parameter is set, then parameter `use.junctions` must also be set.

`use.junctions`
Logical value. This is an alternate parameterization of the `analysis.type` parameter. If `TRUE`, then splice junction loci will be included in the analyses and will be tested for differential usage. If this parameter is set, then parameter `use.exons` must also be set.

`use.novel.junctions`
Logical value. If `TRUE`, then novel splice junctions will not be filtered out prior to analysis.

`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`, `testForDiffUsage`, and `estimateEffectSizes`.

Usage

```
runJunctionSeqAnalyses(sample.files, sample.names, condition,
                       flat.gff.file,
                       outfile.prefix, saveState = FALSE,
                       analysis.type = c("junctionsAndExons","junctionsOnly","
                       use.exons, use.junctions,
                       use.novel.junctions = TRUE,
                       meanCountTestableThreshold = 7.5,
                       nCores = 1,
                       use.covars = NULL,
                       test.formula0 = formula(~ sample + countbin),
                       test.formula1 = formula(~ sample + countbin + condition
                       effect.formula = formula(~ condition + countbin + condi
                       geneLevel.formula = formula(~ condition),
                       gzip.output = TRUE,
                       test.aggregated.genes = FALSE,
```

```
                                fitDispersionsForExonsAndJunctionsSeparately = TRUE,
                                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.
`analysis.type`	Character string. One of "junctionsAndExons", "junctionsOnly", or "exonsOnly". This parameter determines what type of analysis is to be performed. By default JunctionSeq tests both splice junction loci and exonic regions for differential usage (a "hybrid" analysis). This parameter can be used to limit analyses specifically to either splice junction loci or exonic regions.
`use.exons`	Logical value. This is an alternate parameterization of the `analysis.type` parameter. If `TRUE`, then exonic region loci will be included in the analyses and will be tested for differential usage. If this parameter is set, then parameter `use.junctions` must also be set.
`use.junctions`	Logical value. This is an alternate parameterization of the `analysis.type` parameter. If `TRUE`, then splice junction loci will be included in the analyses and will be tested for differential usage. If this parameter is set, then parameter `use.exons` must also be set.
`use.novel.junctions`	Logical value. If `TRUE`, then novel splice junctions will not be filtered out prior to analysis.
`meanCountTestableThreshold`	Numeric value. Features with a total mean normalized count of less than this value will be excluded from 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".

`geneLevel.formula`
For advanced users. The base formula for the model used to estimate total gene-level expression.

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.

`fitDispersionsForExonsAndJunctionsSeparately`
When running a "hybrid" analysis in which both exons and splice junctions are being tested simultaniously, this parameter determines whether a single fitted dispersion model should be fitted for both exons and splice junctions, or if separate fitted dispersions should be calculated for each. By default the dispersions are run separately.

`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.

`testForDiffUsage` *Test Junctions for Differential Junction Usage*

Description

This function runs the hypothesis tests for differential junction usage.

This function is called internally by the `runJunctionSeqAnalyses` function, and thus for most purposes users should not need to call this function directly. It may be useful to advanced users performing non-standard analyses.

Usage

```
testForDiffUsage( 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,
```

```
                      plot.exons = TRUE, plot.junctions = TRUE, plot.novel.junctio
                      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,
                   plot.exons = TRUE, plot.junctions = TRUE, plot.novel.junction
                   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 testForDiffUsage. 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 loci will be included.
only.testable	Logical. If TRUE, only loci with sufficiently high expression to be tested will be included.
plot.exons	Logical. If TRUE, exons will be plotted.
plot.junctions	Logical. If TRUE, splice junctions will be plotted.
plot.novel.junctions	Logical. If TRUE, novel splice junctions will be plotted (if plot.junctions is also TRUE).

`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 loci. 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 locus 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 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 = FALSE, save.VST = FALSE,
                         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 testForDiffUsage. 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