Package 'QoRTs'

May 1, 2017

Version 1.2.37

Date 2017-05-01

Title Quality of RNA-seq Tool

Author Stephen Hartley, PhD

Maintainer Stephen Hartley <QoRTs-contact@list.nih.gov>

Depends R (>= 3.0.2), methods

Suggests MASS, Cairo, DESeq2, edgeR, knitr, BiocStyle, QoRTsExampleData

Enhances png

Description An R toolset for organizing, visualizing, and analyzing RNA-Seq Quality-Control data.

License file LICENSE

VignetteBuilder knitr

URL http://hartleys.github.io/QoRTs/

BugReports https://github.com/hartleys/QoRTs/issues

NeedsCompilation no

R topics documented:

`build.plotter` *Generating plotters*

Description

Generating QC_Plotter objects, which can be used in many of the QoRT utilities to organize samples in various ways to allow for easy comparison and detection of consistent biases and artifacts.

Usage

```
build.plotter.basic(res, plotter.params = list())

build.plotter.colorByGroup(res, plotter.params = list())

build.plotter.colorByLane(res, plotter.params = list())

build.plotter.colorBySample(res, plotter.params = list())

build.plotter.highlightSample(curr.sample,
                              res,
                              plotter.params = list(),
                              merge.offset.outgroup = TRUE)

build.plotter.highlightSample.colorByLane(curr.sample,
                              res,
                              plotter.params = list(),
                              merge.offset.outgroup = TRUE,
                              lane.column.name = "lane.ID")

build.plotter.colorByX(res, color.by.name,
                       color.by.title.name = color.by.name,
                       plotter.params = list())
```

Arguments

res	A QoRT_QC_Results object, created by read.qc.results.data.
curr.sample	A character string. For the sample highlight summary plots, this should be the sample.ID of the sample that is to be highlighted.
merge.offset.outgroup	(For advanced users) A logical value. For the sample highlight plots, determines whether the all lanes that do not include the current sample should be treated as a single "outgroup".
plotter.params	(For advanced users) A named list. Allows you to specify colors, offsets, and other similar patterns. By default these will all be set to reasonable values, however, if you want more control over colors, line-transparency, point plotting characters, or similar, then you can specify a named list. Any parameters that are not specified in the plotter.params list will be left as default. Legal parameters are: • "contrasting.colors": colors to use for contrast. By default these are set to a series of reasonably-contrasting colors. However, if you have too many different categories then it may be hard to tell some colors apart. • "contrasting.pch": point types to use for contrast (see pch in graphical parameters). By default this is set to the basic point types, then following through the upper and lower case letters. • "std.color": Color to use for the "highlighted" replicates. • "std.lines.lty": Line type to use for the "highlighted" replicates. (see lty in graphical parameters) • "std.lines.lwd": Line width to use for the "highlighted" replicates. (see lwd in graphical parameters) • "std.lines.alpha": Alpha transparency value to use on lines for the "highlighted" replicates. Numeric value between 0 and 255. • "std.points.pch": Character to use for points for the "highlighted" replicates. (see pch in graphical parameters) • "std.points.alpha": Alpha transparency value to use on points for the "highlighted" replicates. Numeric value between 0 and 255. • "std.points.color": Color to use for the "highlighted" replicates. • "std.NVC.colors": A named list with elements named "A", "T", "C", and "G", with each element specifying a color. The colors used to indicate each base for the "highlighted" replicates in the nucleotide-rate-by-position plots. • "alt.color": Color to use for the "non-highlighted" replicates. • "alt.lines.lty": Line type to use for the "non-highlighted" replicates. (see lty in graphical parameters) • "alt.lines.lwd": Line width to use for the "non-highlighted" replicates. (see lwd in graphical parameters) • "alt.lines.alpha": Alpha transparency value to use on lines for the "non-highlighted" replicates. Numeric value between 0 and 255. • "alt.points.pch": Character to use for points for the "non-highlighted" replicates. (see pch in graphical parameters) • "alt.points.alpha": Alpha transparency value to use on points for the "non-highlighted" replicates. Numeric value between 0 and 255.

- "alt.NVC.colors": A named list with elements named "A", "T","C", and "G", with each element specifying a color. The colors used to indicate each base for the "non-highlighted" replicates in the nucleotide-rate-by-position plots.
- "show.legend": DEPRECIATED. Currently nonfunctional.

color.by.name
: (For advanced users) A character string. (TODO: document functionality)

color.by.title.name
: (For advanced users) A character string. (TODO: document functionality)

lane.column.name
: The name of the column in the decoder containing the "lane" names.

Value

A QoRT_Plotter reference object used to create QC summary plots. Depending on which plotter is used, samples/lane-bams can be organized by group, sample, lane, or any arbitrary variable found in the decoder.

See Also

read.qc.results.data

Examples

```
data(res,package="QoRTsExampleData");
plotter.basic <- build.plotter.basic(res);
makePlot.insert.size(plotter.basic);

plotter.colorByGroup <- build.plotter.colorByGroup(res);
makePlot.insert.size(plotter.colorByGroup);
makePlot.legend.over("topright",plotter.colorByGroup);

plotter.colorByLane <- build.plotter.colorByLane(res);
makePlot.insert.size(plotter.colorByLane);
makePlot.legend.over("topright",plotter.colorByLane);

plotter.colorBySample <- build.plotter.colorBySample(res);
makePlot.insert.size(plotter.colorBySample);
makePlot.legend.over("topright",plotter.colorBySample);

plotter.HS <- build.plotter.highlightSample("SAMP1",
                                            res);
makePlot.insert.size(plotter.HS);
makePlot.legend.over("topright",plotter.HS);

plotter.HSCBL <- build.plotter.highlightSample.colorByLane("SAMP1",
                                                           res);
makePlot.insert.size(plotter.HSCBL);
makePlot.legend.over("topright",plotter.HSCBL);
```

get.summary.table *Get summary data tables*

Description

Retrieves and compiles a summary data table.

Usage

```
get.summary.table(res, outfile, debugMode);

get.size.factors(res,
     sf.method = c("DESeq2","DESeq2_GEO","TC",
                   "edgeR","edgeR_TMM","edgeR_UQ","edgeR_RLE"),
     outfile, debugMode)
```

Arguments

res	A QoRT_QC_Results object, created by read.qc.results.data.
outfile	Optional. A file name where the table should be written.
sf.method	The size factor method to use. Note that most of these methods (except "TC") are reliant on external packages not included with QoRTs. You will need to install DESeq2 and edgeR to use these methods.
debugMode	Logical. If TRUE, debugging data will be printed to the console.

Details

Returns summary data in table form.

See Also

read.qc.results.data

Examples

```
data(res,package="QoRTsExampleData");
get.summary.table(res);
```

makeMultiPlot *Generating summary multi-plots*

Description

Creates a large multi-frame summary plot, or a report PDF file.

Usage

```
makeMultiPlot.basic(res,
                    outfile,
                    outfile.dir = "./",
                    outfile.prefix = "plot-basic",
                    outfile.ext,
                    plotter.params = list(),
                    plot.device.name = "curr",
                    plotting.device.params = list(),
                    verbose = TRUE,
                    debugMode,
                    rasterize.large.plots,
                    rasterize.medium.plots,
                    raster.height = 1050,
                    raster.width = 1050,
                    separatePlots = FALSE,
                    exclude.autosomes.chrom.rate.plot = TRUE,
                    chromosome.name.style = "UCSC",
                    fig.res = 150, fig.base.height.inches = 7,
                    insertSize.plot.xlim,
                    sequencing.type = c("RNA","Exome","Genome"),
                    nvc.mark.points = TRUE,
                    maxColumns,
                    maxRows,
                    plotList,
                    labelPlots = TRUE,
                    plot = TRUE,
                    ...)

makeMultiPlot.colorByGroup(res,
                    outfile,
                    outfile.dir = "./",
                    outfile.prefix = "plot-colorByGroup",
                    outfile.ext,
                    plotter.params = list(),
                    plot.device.name = "curr",
                    plotting.device.params = list(),
                    verbose = TRUE,
                    debugMode,
                    rasterize.large.plots,
                    rasterize.medium.plots,
                    raster.height = 1050,
                    raster.width = 1050,
                    separatePlots = FALSE,
                    exclude.autosomes.chrom.rate.plot = TRUE,
                    chromosome.name.style = "UCSC",
                    fig.res = 150, fig.base.height.inches = 7,
                    insertSize.plot.xlim,
                    sequencing.type = c("RNA","Exome","Genome"),
                    nvc.mark.points = TRUE,
                    maxColumns,
                    maxRows,
```

```
                        plotList,
                        labelPlots = TRUE,
                        plot = TRUE,
                        ...)

    makeMultiPlot.colorByLane(res,
                        outfile,
                        outfile.dir = "./",
                        outfile.prefix = "plot-colorByLane",
                        outfile.ext,
                        plotter.params = list(),
                        plot.device.name = "curr",
                        plotting.device.params = list(),
                        verbose = TRUE,
                        debugMode,
                        rasterize.large.plots,
                        rasterize.medium.plots,
                        raster.height = 1050,
                        raster.width = 1050,
                        separatePlots = FALSE,
                        exclude.autosomes.chrom.rate.plot = TRUE,
                        chromosome.name.style = "UCSC",
                        fig.res = 150, fig.base.height.inches = 7,
                        insertSize.plot.xlim,
                        sequencing.type = c("RNA","Exome","Genome"),
                        nvc.mark.points = TRUE,
                        maxColumns,
                        maxRows,
                        plotList,
                        labelPlots = TRUE,
                        plot = TRUE,
                        ...)

    makeMultiPlot.colorBySample(res,
                        outfile,
                        outfile.dir = "./",
                        outfile.prefix = "plot-colorByLane",
                        outfile.ext,
                        plotter.params = list(),
                        plot.device.name = "curr",
                        plotting.device.params = list(),
                        verbose = TRUE,
                        debugMode,
                        rasterize.large.plots,
                        rasterize.medium.plots,
                        raster.height = 1050,
                        raster.width = 1050,
                        separatePlots = FALSE,
                        exclude.autosomes.chrom.rate.plot = TRUE,
                        chromosome.name.style = "UCSC",
                        fig.res = 150, fig.base.height.inches = 7,
                        insertSize.plot.xlim,
```

```
                        sequencing.type = c("RNA","Exome","Genome"),
                        nvc.mark.points = TRUE,
                        maxColumns,
                        maxRows,
                        plotList,
                        labelPlots = TRUE,
                        plot = TRUE,
                        ...)

makeMultiPlot.highlightSample(res, curr.sample,
                        outfile,
                        outfile.dir = "./",
                        outfile.prefix = paste0("plot-sampleHL-",curr.sample),
                        outfile.ext,
                        plotter.params = list(),
                        plot.device.name = "curr",
                        plotting.device.params = list(),
                        verbose = TRUE,
                        debugMode,
                        rasterize.large.plots,
                        rasterize.medium.plots,
                        raster.height = 1050,
                        raster.width = 1050,
                        separatePlots = FALSE,
                        exclude.autosomes.chrom.rate.plot = TRUE,
                        chromosome.name.style = "UCSC",
                        fig.res = 150, fig.base.height.inches = 7,
                        insertSize.plot.xlim,
                        sequencing.type = c("RNA","Exome","Genome"),
                        nvc.mark.points = TRUE,
                        maxColumns,
                        maxRows,
                        plotList,
                        labelPlots = TRUE,
                        plot = TRUE,
                        ...)

makeMultiPlot.highlightSample.colorByLane(res, curr.sample,
                        outfile,
                        outfile.dir = "./",
                        outfile.prefix = paste0("plot-sampleHL-coloredByLane-",cur
                        outfile.ext,
                        plotter.params = list(),
                        plot.device.name = "curr",
                        plotting.device.params = list(),
                        verbose = TRUE,
                        debugMode,
                        rasterize.large.plots,
                        rasterize.medium.plots,
                        raster.height = 1050,
                        raster.width = 1050,
                        separatePlots = FALSE,
```

```
                        exclude.autosomes.chrom.rate.plot = TRUE,
                        chromosome.name.style = "UCSC",
                        fig.res = 150, fig.base.height.inches = 7,
                        insertSize.plot.xlim,
                        sequencing.type = c("RNA","Exome","Genome"),
                        nvc.mark.points = TRUE,
                        maxColumns,
                        maxRows,
                        plotList,
                        labelPlots = TRUE,
                        plot = TRUE,
                        ...)

  makeMultiPlot.withPlotter(plotter,
                        res = plotter$res,
                        outfile,
                        outfile.dir = "./",
                        outfile.prefix = "plot-custom",
                        outfile.ext,
                        plotter.params = list(),
                        plot.device.name = "curr",
                        plotting.device.params = list(),
                        verbose = TRUE,
                        debugMode,
                        rasterize.large.plots,
                        rasterize.medium.plots,
                        raster.height = 1050,
                        raster.width = 1050,
                        separatePlots = FALSE,
                        exclude.autosomes.chrom.rate.plot = TRUE,
                        chromosome.name.style = "UCSC",
                        fig.res = 150, fig.base.height.inches = 7,
                        insertSize.plot.xlim,
                        sequencing.type = c("RNA","Exome","Genome"),
                        nvc.mark.points = TRUE,
                        maxColumns,
                        maxRows,
                        plotList,
                        labelPlots = TRUE,
                        plot = TRUE,
                        ...)
```

Arguments

`res`	A `QoRT_QC_Results` object, created by `read.qc.results.data`.
`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`curr.sample`	A character string. For the sample highlight summary plots, this should be the sample.ID of the sample that is to be highlighted.
`outfile`	The output file can be specified in two ways: first, by setting the outfile parameter, which should be the full path for the output file. Alternatively, the file path

will be the concatenation of the three parameters: outfile.dir, outfile.prefix, and outfile.ext.

If plot.device.name == "curr", then all the outfile parameters will be nonfunctional.

`outfile.dir` A prefix to be appended to the output filename. Usually the path to the destination directory. By default, this will be the current working directory.

`outfile.prefix`
A second prefix to be appended to the output filename, after outfile.dir. This is usually the name of the file, without the file extension. By default, a reasonable file name will be selected.

`outfile.ext` The file extension. By default, this will be set to match the selected graphics device.

`plotter.params`
Additional parameters (advanced) used in creation of the Plotter objects. See build.plotter.

`plot.device.name`
The method to use to save plots. Several formats and devices are supported:

- `"curr"` (default) Do not create output files. Plot directly to the current or default device.
- `"png"` for standard png compression.
- `"CairoPNG"` for png compression using package Cairo. Note that this requires the package Cairo.
- `"tiff"` for the tiff device.
- `"jpeg"` for the jpeg device (*not recommended!*)
- `"svg"` for the vector svg device. Note that for all vector devices, by default, certain large plots will be rasterized if the `png` package is found. If not, the function will still work, but output files may be very large and may have trouble rendering and printing.
- `"pdf"` for a multi-page pdf report.
- `"CairoPDF"` for a multi-page pdf report using package Cairo. Note that this requires the package Cairo.

`plotting.device.params`
A named list of parameters to be passed to the graphics device. For example:

- `width = 2000`
- `height = 2000`
- `units = "px"`

Reasonable values for height, width, and pointsize will be chosen by default. Generally all raster plots will be set to 1000 by 1000 pixels, and all vector plots will be set to 7 inches by 7 inches.

`verbose` Logical. If TRUE, more information will be printed to the console.

`debugMode` Logical. If TRUE, debugging data will be printed to the console.

`rasterize.large.plots`
Logical. If TRUE, then if the currently selected plotting device is a vector device (or the "curr" device), then certain large plots will have their plotting areas rasterized. The axis labels, titles, and text will all remain vectorized, only the plotting areas will be flattened. *Note that this requires the png package.*

By default, this parameter will be set to TRUE when a vector device is selected.

`rasterize.medium.plots`
: Logical. as rasterize.large.plots, but applies to moderately-large plots. By default, this parameter will be set to TRUE for pdf/CairoPDF output only.

`raster.height`
: Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.

`raster.width`
: Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels. Double-pane plots will be twice this width.

`separatePlots`
: Logical. If TRUE, then separate image files will be saved for each individual plot, rather than saving one large multi-pane plot. Note that this is not compatible with the "curr" device. Also note: if this parameter is set to TRUE, then the output files will be saved using the outfile.dir, outfile.prefix and outfile.ext parameters. The "outfile" parameter cannot be set. The files will be saved to the contatenation of outfile.dir, outfile.prefix, the name of the plot type, and then outfile.ext.

`exclude.autosomes.chrom.rate.plot`
: A logical value indicating whether to exclude autosomes from the plot. See `makePlot.chrom.type.rates`

`chromosome.name.style`
: A string value indicating the style of the chromosome names, and also how to split up the categories. See `makePlot.chrom.type.rates`

`fig.res`
: Numeric value. The number of pixels per "inch" (for raster devices only). For some plotting devices the figure height will be in pixels not inches, and will equal this value multiplied by the fig.base.height.inches value.

`fig.base.height.inches`
: Numeric value. The base height, in inches, of each sub-figure in the plot. This will be equal to the height for vector devices, or used to calculate the height in pixels using the fig.res value (see above).

`insertSize.plot.xlim`
: A numeric vector of length 2. The x-axis limits for the insert size plot. By default QoRTs will attempt to find reasonable values for this, but there are always situations where the default behavior is not ideal. Using this parameter you can set it explicitly.

`nvc.mark.points`
: Logical. If TRUE, then points should be marked with shapes on the NVC plots.

`sequencing.type`
: The type of sequencing data being analyzed. This only changes the default plot set, which can be overriden with the `plotList` parameter.

`maxColumns`
: If set, QoRTs will attempt to create a multiplot that has (at most) maxColumns columns. Extra rows will be added to fit all the plots, as needed.

`maxRows`
: If set, QoRTs will attempt to create a multiplot that has (at most) maxRows rows. Extra columns will be added to fit all the plots as needed. To set the number of rows and columns manually, you can set both maxColumns and maxRows.

`plotList`
: The list of desired plots.

`labelPlots`
: Logical. If TRUE, then label each plot with a letter.

`plot`
: Logical. If FALSE, do not create any plots. (In other words, do nothing but test the ability to create plots).

`...`
: Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

Produces large, multi-frame summary plots, with a large number of different plots and graphs.

For devices pdf and CairoPDF, this function will produce a multi-page document suitable for printing, with (by default) 6 panels on each page.

Note that for all vector devices (svg, pdf, and CairoPDF), by default, QoRTs will attempt to load the `png` package. If this package is found, then certain large plots (the NVC plots and the cumulative diversity plots) will be rasterized within the plotting area. The labels and text will still be vectorized. If the `png` package is not installed, the function will still work, but output files may be very large and may have trouble rendering and printing.

Rasteration of large plots can be turned off via the rasterize.large.plots option.

See Also

`build.plotter`

Examples

```
## Not run:
  data(res,package="QoRTsExampleData");
  makeMultiPlot.colorByGroup(res);

## End(Not run)
```

---

makeMultiPlot.highlightSample.all

Generating sample highlight multi-plots

Description

Generates multiple sample highlight summary plots, one for every sample.

Usage

```
makeMultiPlot.highlightSample.all(res, outfile.dir = "./",
              plotter.params = list(),
              plot.device.name = "png",
              plotting.device.params = list(),
              verbose = TRUE,
              debugMode ,
              rasterize.large.plots,
              rasterize.medium.plots,
              raster.height = 1050,
              raster.width = 1050,
              exclude.autosomes.chrom.rate.plot = TRUE,
              chromosome.name.style = "UCSC",
              fig.res = 150, fig.base.height.inches = 7,
              insertSize.plot.xlim,
              sequencing.type = c("RNA","Exome","Genome"),
              maxColumns,
              maxRows,
```

```
                plotList,
                labelPlots=TRUE,
                ...)

makeMultiPlot.highlightSample.colorByLane.all(res,
                outfile.dir = "./",
                plotter.params = list(),
                plot.device.name = "png",
                plotting.device.params = list(),
                verbose = TRUE,
                debugMode ,
                rasterize.large.plots,
                rasterize.medium.plots,
                raster.height = 1050,
                raster.width = 1050,
                exclude.autosomes.chrom.rate.plot = TRUE,
                chromosome.name.style = "UCSC",
                fig.res = 150, fig.base.height.inches = 7,
                insertSize.plot.xlim,
                sequencing.type = c("RNA","Exome","Genome"),
                maxColumns,
                maxRows,
                plotList,
                labelPlots=TRUE,
                ...)
```

Arguments

`res`	A `QoRT_QC_Results` object, created by `read.qc.results.data`.
`outfile.dir`	A file prefix, used for all output files. Usually the directory to which you want all files to be written.
`plotter.params`	Additional parameters (advanced) used in creation of the Plotter objects. See build.plotter.
`plot.device.name`	The method to use to save plots. Can be one of: • `"png"` for standard png compression, • `"CairoPNG"` for png compression using package Cairo. Note that this requires the package Cairo.
`plotting.device.params`	A named list of parameters to be passed to the graphics device. For example: • `"width = 2000"` Reasonable values for height, width, and pointsize will be chosen by default.
`verbose`	Logical. If TRUE, more info will be printed to the console.
`debugMode`	Logical. If TRUE, debugging data will be printed to the console.
`rasterize.large.plots`	Logical. If TRUE, then if the currently selected plotting device is a vector device (or the "curr" device), then certain large plots will have their plotting areas rasterized. The axis labels, titles, and text will all remain vectorized, only the plotting areas will be flattened. *Note that this requires the png package.* By default, this parameter will be set to TRUE when a vector device is selected.

`rasterize.medium.plots`
: Logical. as rasterize.large.plots, but applies to moderately-large plots. By default, this parameter will be set to TRUE for pdf/CairoPDF output only.

`raster.height`
: Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.

`raster.width`
: Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels. Double-pane plots will be twice this width.

`exclude.autosomes.chrom.rate.plot`
: A logical value indicating whether to exclude autosomes from the plot. See `makePlot.chrom.type.rates`

`chromosome.name.style`
: A string value indicating the style of the chromosome names, and also how to split up the categories. See `makePlot.chrom.type.rates`

`fig.res`
: Numeric value. The number of pixels per "inch" (for raster devices only). For some plotting devices the figure height will be in pixels not inches, and will equal this value multiplied by the fig.base.height.inches value.

`fig.base.height.inches`
: Numeric value. The base height, in inches, of each sub-figure in the plot. This will be equal to the height for vector devices, or used to calculate the height in pixels using the fig.res value (see above).

`insertSize.plot.xlim`
: A numeric vector of length 2. The x-axis limits for the insert size plot. By default QoRTs will attempt to find reasonable values for this, but there are always situations where the default behavior is not ideal. Using this parameter you can set it explicitly.

`sequencing.type`
: The type of sequencing data being analyzed. This only changes the default plot set, which can be overriden with the `plotList` parameter.

`maxColumns`
: If set, QoRTs will attempt to create a multiplot that has (at most) maxColumns columns. Extra rows will be added to fit all the plots, as needed.

`maxRows`
: If set, QoRTs will attempt to create a multiplot that has (at most) maxRows rows. Extra columns will be added to fit all the plots as needed. To set the number of rows and columns manually, you can set both maxColumns and maxRows.

`plotList`
: The list of desired plots.

`labelPlots`
: Logical. If TRUE, then label each plot with a letter.

`...`
: Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

Generates a sample highlight plot for each sample in the dataset.

See Also

`build.plotter`, `makeMultiPlot`

Examples

```
## Not run:
data(res,package="QoRTsExampleData");
  makeMultiPlot.highlightSample.all(res);
  makeMultiPlot.highlightSample.colorByLane.all(res);

## End(Not run)
```

---

makePlot.all.std *Generating all default plots*

Description

Saves MANY compiled QC plots for the given dataset.

Usage

```
makeMultiPlot.all(res, outfile.dir = "./",
                  plotter.params = list(),
                  plot.device.name = "png",
                  plotting.device.params = list(),
                  debugMode,
                  rasterize.large.plots,
                  rasterize.medium.plots,
                  raster.height = 1050,
                  raster.width = 1050,
                  exclude.autosomes.chrom.rate.plot = TRUE,
                  chromosome.name.style = "UCSC",
                  fig.res = 150, fig.base.height.inches = 7,
                  insertSize.plot.xlim ,
                  sequencing.type = c("RNA","Exome","Genome"),
                  maxColumns,
                  maxRows,
                  plotList,
                  labelPlots = TRUE,
                  ...)
```

Arguments

`res`	A `QoRT_QC_Results` object, created by `read.qc.results.data`.
`outfile.dir`	A file prefix, used for all output files. Usually the directory to which you want all files to be written.
`plotter.params`	Additional (advanced) parameters used in creation of the Plotter objects. See build.plotter.
`plot.device.name`	The method to use to save plots. Can be one of: • `"png"` for standard png compression, • `"CairoPNG"` for png compression using package Cairo. Note that this requires the package Cairo.

`plotting.device.params`
: A named list of parameters to be passed to the graphics device. For example:

  - `"width = 2000"`

  Reasonable values for height, width, and pointsize will be chosen by default.

`debugMode`
: Logical. If TRUE, debugging data will be printed to the console.

`rasterize.large.plots`
: Logical. If TRUE, then if the currently selected plotting device is a vector device (or the "curr" device), then certain large plots will have their plotting areas rasterized. The axis labels, titles, and text will all remain vectorized, only the plotting areas will be flattened. *Note that this requires the png package.*

  By default, this parameter will be set to TRUE when a vector device is selected.

`rasterize.medium.plots`
: Logical. as rasterize.large.plots, but applies to moderately-large plots. By default, this parameter will be set to TRUE for pdf/CairoPDF output only.

`raster.height`
: Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.

`raster.width`
: Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels. Double-pane plots will be twice this width.

`exclude.autosomes.chrom.rate.plot`
: A logical value indicating whether to exclude autosomes from the plot. See `makePlot.chrom.type.rates`

`chromosome.name.style`
: A string value indicating the style of the chromosome names, and also how to split up the categories. See `makePlot.chrom.type.rates`

`fig.res`
: Numeric value. The number of pixels per "inch" (for raster devices only). For some plotting devices the figure height will be in pixels not inches, and will equal this value multiplied by the fig.base.height.inches value.

`fig.base.height.inches`
: Numeric value. The base height, in inches, of each sub-figure in the plot. This will be equal to the height for vector devices, or used to calculate the height in pixels using the fig.res value (see above).

`insertSize.plot.xlim`
: A numeric vector of length 2. The x-axis limits for the insert size plot. By default QoRTs will attempt to find reasonable values for this, but there are always situations where the default behavior is not ideal. Using this parameter you can set it explicitly.

`sequencing.type`
: The type of sequencing data being analyzed. This only changes the default plot set, which can be overriden with the `plotList` parameter.

`maxColumns`
: If set, QoRTs will attempt to create a multiplot that has (at most) maxColumns columns. Extra rows will be added to fit all the plots, as needed.

`maxRows`
: If set, QoRTs will attempt to create a multiplot that has (at most) maxRows rows. Extra columns will be added to fit all the plots as needed. To set the number of rows and columns manually, you can set both maxColumns and maxRows.

`plotList`
: The list of desired plots.

`labelPlots`
: Logical. If TRUE, then label each plot with a letter.

`...`
: Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

Saves MANY compiled QC plots for the given dataset, calling all standard variants of `makeMultiPlot`.

See Also

`read.qc.results.data`, `build.plotter`, `makeMultiPlot`

Examples

```
## Not run:
data(res,package="QoRTsExampleData");
  makeMultiPlot.all(res, outfile.dir = "./");

## End(Not run)
```

---

makePlot.biotype.rates

Plot Biotype rates

Description

Plots counts for each gene biotype. This plot is only useful and informative when QoRTs is run on a GTF file that contains "gene_biotype" tags.

Usage

```
makePlot.biotype.rates(plotter,
            plot.rates = TRUE,
            count.type = c("all","unambigOnly"),
            log.y = TRUE,
            return.table = FALSE,
            debugMode = DEFAULTDEBUGMODE,
            singleEndMode = plotter$res@singleEnd,
            showTypes,
            plot = TRUE,
            ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`plot.rates`	A logical value. If `TRUE`, then reads/read-pairs per million should be plotted, rather than raw counts. Default is `TRUE`.
`count.type`	A logical value. If `TRUE`, then both ambiguous and unambiguous reads will be counted. Otherwise only unambiguous reads will be counted.
`log.y`	A logical value indicating that the y-axis should be log-scaled.
`return.table`	Logical. If `TRUE`, then return a data table.
`debugMode`	Logical. If TRUE, debugging data will be printed to the console.
`singleEndMode`	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

showTypes	Character vector. An optional list of biotypes to include in the plot. By default all observed biotypes will be plotted.
plot	Logical. Default is TRUE. If FALSE, then do NOT plot any results, instead just return a data table.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

x-axis: Gene "BioTypes", from the annotation GTF.

y-axis: Total read counts across all genes in the BioType, measured either in reads or reads-per-million.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.biotype.rates(plotter);
```

---

makePlot.chrom.type.rates

Chromosome type rate plot

Description

Plots the number or percent of read-pairs falling on each type of chromosome.

Usage

```
makePlot.chrom.type.rates(plotter,
                          plot.rates = TRUE,
                          chromosome.name.style = "UCSC",
                          exclude.autosomes = FALSE,
                          chrom.norm.factors = NULL,
                          custom.chromosome.style.def.function = NULL,
                          return.table = FALSE,
                          debugMode, singleEndMode,
                          plot = TRUE,
                          ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
plot.rates	A logical value indicating whether to plot percent of the total (for each bam file), rather than read-counts.
chromosome.name.style	A string value indicating the style of the chromosome names, and also how to split up the categories. There are 4 legal options:

- "UCSC": The default. Chromosomes are named: "chr1,chr2,...,chrX,chrY,chrXY,chrM". There are 6 categories: autosome, X, Y, XY, mitochondrial, and other.
- "ENSEMBL": Chromosomes are named: "1,2,...,X,Y,XY,MT". There are 6 categories: autosome, X, Y, XY, mitochondrial, and other.
- "UCSC_WITH_ERCC": As UCSC, but there is an additional category, which contains all chromosomes that begin with the text "ERCC".
- "ENSEMBL_WITH_ERCC": As ENSEMBL, but there is an additional category, which contains all chromosomes that begin with the text "ERCC".

`chrom.norm.factors`
: (Advanced users)

`exclude.autosomes`
: A logical value indicating whether to exclude autosomes from the plot.

`custom.chromosome.style.def.function`
: (For advanced users) If your chromosomes do not match any of the above styles, then you can set your own chromosome style by handing this option a function. The function must take one argument. When handed NULL, it must return a list of all legal categories. When handed a single chromosome name, it must return one of those categories.

`return.table`
: A Logical value. If TRUE, the function will return a table containing the plotted data.

`debugMode`
: Logical. If TRUE, debugging data will be printed to the console.

`singleEndMode`
: Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

`plot`
: Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

`...`
: Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

For each sample-run, this function plots the number of read-pairs mapping to each category of chromosome.

Value

By default, this function returns nothing. If the return.table parameter is TRUE, then it returns a data.frame with the data that was plotted.

See Also

`build.plotter`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.chrom.type.rates(plotter);
```

`makePlot.cigarLength.distribution` *Plot the length distribution of a given cigar operation*

Description

Plots the length distribution of a given cigar operation

Usage

```
makePlot.cigarLength.distribution(plotter, op,
                                  r2.buffer = NULL,
                                  perMillion = TRUE,
                                  log.x = FALSE,
                                  log.y = FALSE,
                                  debugMode, singleEndMode,
                                  rasterize.plotting.area = FALSE, raster.height =
                                  plot = TRUE,
                                  ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`op`	A character string naming which cigar op to plot. Must be one of the following: • `"SoftClip"` Soft Clip (Cigar Op "S") • `"HardClip"` Hard Clip (Cigar Op "H") • `"Del"` Deletion from reference (Cigar Op "D") • `"Ins"` Insertion from reference (Cigar Op "I") • `"Pad"` Padding (Cigar Op "P") • `"Splice"` Splice Junction (Cigar Op "N") • `"Aln"` Aligned to reference (Cigar Op "M") Note that makePlot.cigarOp.byCycle(plotter,"SoftClip") gives identical results as makePlot.clipping, although the default value for the `rate.per.million` argument is different.
`r2.buffer`	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
`perMillion`	A logical value indicating whether or not to scale the y axis to rate-per-million-reads, rather than rate-per-read. Some people may find the results more readable this way, even though the plots themselves will appear the same.
`log.x`	A logical value indicating whether or not to log-scale the x axis.
`log.y`	A logical value indicating whether or not to log-scale the y axis.
`debugMode`	Logical. If TRUE, debugging data will be printed to the console.
`singleEndMode`	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

`rasterize.plotting.area`
: Logical. If `TRUE`, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is `FALSE`.

`raster.height`
: Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the height of the rasterized plot, in pixels.

`raster.width`
: Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the width of the rasterized plot, in pixels.

`plot`
: Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

`...`
: Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

x-axis: Length of the cigar operation.

y-axis: Frequency of the given length.

See Also

`build.plotter`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.cigarLength.distribution(plotter, op = "Del");
makePlot.cigarLength.distribution(plotter, op = "Ins");
```

---

makePlot.cigarOp.byCycle

Plot Cigar Operator Rate

Description

Plots the rate at which the given CIGAR operator occurs, by read cycle.

Usage

```
makePlot.cigarOp.byCycle(plotter,
          op=c("SoftClip","Del","Ins","HardClip","Pad","Splice","Aln"),
          r2.buffer = NULL,
          rate.per.million = TRUE,
          use.readLength.denominator = TRUE,
          debugMode,
          singleEndMode,
          rasterize.plotting.area = FALSE,
          raster.height = 1000,
          raster.width = 1000,
          plot = TRUE,
          ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
op	A character string naming which cigar op to plot. Must be one of the following: • "SoftClip" Soft Clip (Cigar Op "S") • "HardClip" Hard Clip (Cigar Op "H") • "Del" Deletion from reference (Cigar Op "D") • "Ins" Insertion from reference (Cigar Op "I") • "Pad" Padding (Cigar Op "P") • "Splice" Splice Junction (Cigar Op "N") • "Aln" Aligned to reference (Cigar Op "M") Note that makePlot.cigarOp.byCycle(plotter,"SoftClip") gives identical results as makePlot.clipping, although the default value for the rate.per.million argument is different.
r2.buffer	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
rate.per.million	A logical value indicating whether or not to scale the y axis to rate-per-million-reads, rather than rate-per-read. Some people may find the results more readable this way, even though the plots themselves will appear the same.
use.readLength.denominator	Logical. If TRUE, use the read-length counts as the denominator when calculating op rates. This is only relevant if operating on trimmed reads, where the read length is variable.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
rasterize.plotting.area	Logical. If TRUE, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is FALSE.
raster.height	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the height of the rasterized plot, in pixels.
raster.width	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the width of the rasterized plot, in pixels.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

x-axis: The read cycle (ie. the base-pair position in the read).

y-axis: The rate at which the bases at the given read-cycle is clipped off.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.cigarOp.byCycle(plotter, op = "Del");
makePlot.cigarOp.byCycle(plotter, op = "Ins");
makePlot.cigarOp.byCycle(plotter, op = "Splice");
```

---

makePlot.clipping *Plot Alignment Clipping*

Description

Plots the rate at which the aligner soft-clips off portions of aligned reads.

Usage

```
makePlot.clipping(plotter,
                  rate.per.million = FALSE,
                  use.readLength.denominator = TRUE,
                  r2.buffer,
                  debugMode,
                  singleEndMode,
                  rasterize.plotting.area = FALSE,
                  raster.height = 1000,
                  raster.width = 1000,
                  plot= TRUE, ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
rate.per.million	A logical value indicating whether or not to scale the y axis to rate-per-million-reads, rather than rate-per-read. Some people may find the results more readable this way, even though the plots themselves will appear the same.
use.readLength.denominator	Logical. If TRUE, then use the total number of reads with at least the given position's length as the denominator of the rate. This is only relevant when the dataset has variable-length trimming prior to alignment.
r2.buffer	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

`rasterize.plotting.area`
: Logical. If `TRUE`, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is `FALSE`.

`raster.height`
: Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the height of the rasterized plot, in pixels.

`raster.width`
: Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the width of the rasterized plot, in pixels.

`plot`
: Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

`...`
: Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

x-axis: The read cycle (ie. the base-pair position in the read).

y-axis: The rate at which the bases at the given read-cycle is clipped off.

See Also

`build.plotter`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.clipping(plotter)
```

---

makePlot.dropped.rates

Read Drop Plot

Description

Plots the rates at which reads are dropped from analysis for various causes.

Usage

```
makePlot.dropped.rates(plotter, dropAlwaysZeroRows = FALSE,
                       debugMode, singleEndMode,
                       plot = TRUE,
                       ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
dropAlwaysZeroRows	Logical. If TRUE, drop-reasons that never occur in the dataset will not be plotted. This often cleans up the plot somewhat, since in many production pipelines reads that fail many of the filtering steps may have already been filtered out.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

For each bam file, this function plots the rates and reasons for reads being dropped from QC analysis.

Note that in the example dataset reads were never dropped. This is a consequence of the preprocessing steps in the example pipeline.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.dropped.rates(plotter)
```

---

makePlot.gc *Plot GC content*

Description

Plots GC content.

Usage

```
makePlot.gc(plotter, plot.medians = NULL, plot.means = TRUE,
            plotRate = FALSE, byPair = FALSE, useFQ = FALSE,
            debugMode, singleEndMode,
            rasterize.plotting.area = FALSE, raster.height = 1000, raster.widt
            plot = TRUE,
            ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
plot.medians	A logical value indicating whether or not to plot the median average GC content for each bam file at the bottom of the plot. Overrides plot.means.
plot.means	A logical value indicating whether or not to plot the mean average GC content for each bam file at the bottom of the plot.
plotRate	A logical value indicating whether or not the X-axis should be the raw number of nucleotides that are G/C, vs the rate G/C.
byPair	Logical. If FALSE, GC content rates will be calculated for all reads individually. If TRUE, GC content rates will be calculated for all read-pairs. Note that when the insert size is small, the paired plot can sometimes appear jagged, as completely-overlapped read-pairs will always have an even number of G/C nucleotides.
useFQ	Logical. If TRUE, plot the G/C rate from the unaligned FASTQ data. This requires that the FASTQ data was supplied to QoRTs in the QoRTs QC step.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
rasterize.plotting.area	Logical. If TRUE, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is FALSE.
raster.height	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the height of the rasterized plot, in pixels.
raster.width	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the width of the rasterized plot, in pixels.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

x-axis: Percent of the read-pairs that is made up of G's or C's.

y-axis: Rate at which the given percent occurs.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.gc(plotter)
```

`makePlot.gene.assignment.rates` *Gene assignment rate plot*

Description

Plots the rate at which reads are assigned into various categories.

Usage

```
makePlot.gene.assignment.rates(plotter,  debugMode, singleEndMode,
                               plot = TRUE, ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`debugMode`	Logical. If TRUE, debugging data will be printed to the console.
`singleEndMode`	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
`plot`	Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

For each bam-file, this function plots the rate (y-axis) for which the bam-file's read-pairs are assigned to the given categories.

The categories are:

- *Unique Gene*: The read-pair overlaps with the exonic segments of one and only one gene. For many downstream analyses tools, such as DESeq, DESeq2, and EdgeR, only read-pairs in this category are used.
- *Ambig Gene*: The read-pair overlaps with the exons of more than one gene.
- *No Gene*: The read-pair does not overlap with the exons of any annotated gene.
- *No Gene, Intronic*: The read-pair does not overlap with the exons of any annotated gene, but appears in a region that is bridged by an annotated splice junction.
- *No Gene, 1kb from gene*: The read-pair does not overlap with the exons of any annotated gene, but is within 1 kilobase from the nearest annotated gene.
- *No Gene, 10kb from gene*: The read-pair does not overlap with the exons of any annotated gene, but is within 10 kilobases from the nearest annotated gene.
- *No Gene, middle of nowhere*: The read-pair does not overlap with the exons of any annotated gene, and is *more* than 10 kilobases from the nearest annotated gene.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.gene.assignment.rates(plotter)
```

---

makePlot.gene.cdf *Plot the cumulative gene diversity curve*

Description

Plots the cumulative gene diversity curve

Usage

```
makePlot.gene.cdf(plotter,
                  sampleWise = FALSE,
                  plot.intercepts = TRUE,
                  label.intercepts = FALSE,
                  debugMode,
                  rasterize.plotting.area = FALSE,
                  raster.height = 1000,
                  raster.width = 1000,
                  singleEndMode,
                  plot = TRUE,
                  ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
sampleWise	A logical value indicating whether to compile each sample together across all runs.
plot.intercepts	A logical value indicating whether or not to plot the intercepts with the round numbers on the x-axis, in dotted lines.
label.intercepts	A logical value indicating whether or not to label the intercepts. No effect if plot.intercepts is FALSE.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
rasterize.plotting.area	Logical. If TRUE, the plotting area will be written to a temp png file then drawn to the current device as a raster image. This option is generally preferred for vector devices, because NVC plots can be very large when drawn in vector format. *Note: this requires the* png *package!*
raster.height	Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.

`raster.width` Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels.

`singleEndMode` Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

`plot` Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

`...` Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

For each bam-file, this function plots the cumulative gene diversity. For each bam-file, the genes are sorted by read-count. Then, a cumulative function is calculated for the percent of the total proportion of reads as a function of the number of genes. Intercepts are plotted as well, showing the cumulative percent for 1 gene, 10 genes, 100 genes, 1000 genes, and 10000 genes.

So, for example, across all the bam-files, around 50 to 55 percent of the read-pairs were found to map to the top 1000 genes. Around 20 percent of the reads were found in the top 100 genes. And so on.

This can be used as an indicator of whether a large proportion of the reads stem from of a small number of genes.

Note that this is restricted to only the reads that map to a single unique gene. Reads that map to more than one gene, or that map to intronic or intergenic areas are ignored.

See Also

`build.plotter`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.gene.cdf(plotter)
```

---

makePlot.genebody.coverage

Plot Gene-Body coverage distribution

Description

Plots Gene-Body coverage distribution

Usage

```
makePlot.genebody(plotter,
                  geneset = c("Overall","90-100","75-90","50-75","0-50"),
                  avgMethod = c("TotalCounts", "AvgPercentile"),
                  plot.medians,
                  plot.means = TRUE,
```

```
                    debugMode,
                    singleEndMode,
                    rasterize.plotting.area = FALSE, raster.height = 1000, raster.
                    plot = TRUE,
                    ... )

  #NOTE: The preferred method to access the below functions
  #  is to use makeplot.genebody and set:
  #  avgMethod = "TotalCounts"
  #DEPRECIATED:
  makePlot.genebody.coverage(plotter, plot.medians,
                        plot.means = TRUE,
                        debugMode, singleEndMode,
                        rasterize.plotting.area = FALSE, raster.height = 1000, ra
                        plot = TRUE,
                        ...)
  #DEPRECIATED:
  makePlot.genebody.coverage.UMQuartile(plotter, plot.medians,
                        plot.means = TRUE,
                        debugMode, singleEndMode,
                        rasterize.plotting.area = FALSE, raster.height = 1000, ra
                        plot = TRUE,
                        ...)
  #DEPRECIATED:
  makePlot.genebody.coverage.lowExpress(plotter, plot.medians,
                        plot.means = TRUE,
                        debugMode, singleEndMode,
                        rasterize.plotting.area = FALSE, raster.height = 1000, ra
                        plot = TRUE,
                        ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`geneset`	The set of genes to plot the gene-body coverage over. Genes are grouped into four quantiles by their by their total read counts: the 90-100 quantile, the 75-90 quantile, the 50-75 quantile (or "upper-middle quartile"), and the 0-50 quantile. By default it will plot the overall gene-body coverage across all genes.
`avgMethod`	The method used to generate average gene body coverage. The "TotalCounts" (default) method simply takes the sum of each bin across all genes in the geneset, then for each replicate normalizes the coverage into frequencies. The "AvgPercentile" calculates the coverage distribution across each percentile bin and normalizes the counts so they add up to 1, FOR EACH GENE, and THEN averages those values across all genes in the geneset. The latter variant is experimental. the idea was for it to reduce the impact of highly-expressed genes and gain a better estimate of the degree of 5'/3' bias.
`plot.medians`	A logical value indicating whether or not to plot the median average for each bamfile at the bottom of the plot. Overrides `plot.means`.
`plot.means`	A logical value indicating whether or not to plot the mean average for each bamfile at the bottom of the plot.

`debugMode` Logical. If TRUE, debugging data will be printed to the console.

`singleEndMode`
Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

`rasterize.plotting.area`
Logical. If `TRUE`, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is `FALSE`.

`raster.height`
Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the height of the rasterized plot, in pixels.

`raster.width` Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the width of the rasterized plot, in pixels.

`plot` Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

`...` Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

x-axis: Gene-body quantile. By default this is broken up into 40 quantiles containing 2.5

y-axis: Rate at which reads falls into the given quantile of the genes' bodies.

`makePlot.genebody.coverage` plots the gene body coverage across all genes. `makePlot.genebody.cover` plots the gene body coverage across the genes that fall in the upper-middle quartile of total expression for each sample-run (excluding genes with zero reads). `makePlot.genebody.coverage.lowExpress` plots the gene body coverage across the genes that fall in the lower two quartiles of total expression for each sample-run (excluding genes with zero reads).

See Also

`build.plotter`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.genebody(plotter, geneset = "Overall");
makePlot.genebody(plotter, geneset = "90-100");
makePlot.genebody(plotter, geneset = "50-75");
```

---

makePlot.insert.size

Plot Insert Size Distribution.

Description

Plots the distribution of the size of the region covered by the two paired reads.

Usage

```
makePlot.insert.size(plotter, calc.rate = TRUE, pct.cutoff = 0.98,
                     plot.medians = TRUE, plot.means = NULL,
                     debugMode, singleEndMode, xlim = NULL,
                     rasterize.plotting.area = FALSE, raster.height = 1000, raster
                     plot = TRUE,
                     ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
calc.rate	A logical value indicating whether or not to calculate and plot the rate at which each insert size occurs as the y-axis. If this is set to false, it will instead plot the total number of times each insert size occurs.
pct.cutoff	A numeric value between 0 and 1, indicating the quantile within which to limit the x-axis. Generally the default value of 0.98 is perfectly usable.
plot.means	A logical value indicating whether or not to plot the mean average for each bamfile at the bottom of the plot.
plot.medians	A logical value indicating whether or not to plot the median average for each bamfile at the bottom of the plot. Overrides plot.means.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
xlim	Numeric vector of length 2. Sets the x-axis limits. By default QoRTs will attempt to autodetect reasonable values for this, but there are always cases where you want to set this explicitly.
rasterize.plotting.area	Logical. If TRUE, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is FALSE.
raster.height	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the height of the rasterized plot, in pixels.
raster.width	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the width of the rasterized plot, in pixels.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

x-axis: The insert size. This is the genomic distance from the start of one read to the end of the other. In other words, the size of the full region covered by the paired reads.

y-axis: The rate at which the given insert size occurs, for each bamfile.

Note: The insert size is calculated by using the alignment as well as the provided gene/splice-junction annotation.

1. If the paired reads align to overlapping regions of the reference genome, then the insert size can be calculated exactly. This is NOT dependant on the annotation. If the two reads align inconsistantly (for example, one read showing a splice junction and the other not), then the read is ignored.
2. If the paired reads do NOT overlap, then the annotation information is required. Using the full set of all known splice junctions that lie between the inner alignment endpoints for the two reads, the shortest possible path from the two endpoints is found. In some rare cases this may underestimate the insert size, if the actual insert does not take the minimal path, but this is rare. In other (somewhat more common) cases this may overestimate the insert size, when the region between the paired reads bridges an unannotated splice junction.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.insert.size(plotter);
```

---

makePlot.legend.box

Plot legend

Description

Plots the universal legend for a given plotter object.

Usage

```
makePlot.legend.box(plotter, debugMode, singleEndMode,
                    cex.legend, ncol,
                    plot = TRUE,
                    ...)
makePlot.legend.over(position,
                  plotter,
                  debugMode ,
                  singleEndMode,
                  ncol = NULL,
                  plot = TRUE,
                  ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
position	For makePlot.legend.over, a character string indicating the location where you want the legend to appear. This is passed to legend, and can be any keyword allowed by xy.coords. For example: "top", "topleft","bottomright", etc.
debugMode	Logical. If TRUE, debugging data will be printed to the console.

`singleEndMode`
: Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

`cex.legend`
: Numeric. The cex expansion factor passed to `legend`. By default the cex is autofitted to fill the available space.

`ncol`
: Integer value. The number of columns in the legend. By default this will be selected automatically depending on the number of items in the legend.

`plot`
: Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

`...`
: Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

makePlot.legend.box creates a new plot (opening the next graphics frame), and writes a small description of the given plotter type along with plotting a color-coded legend (if applicable).

makePlot.legend.over adds a legend to the current plotting frame.

See Also

`build.plotter`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
#Add a legend to an existing plot:
makePlot.strandedness.test(plotter);
makePlot.legend.over("topright", plotter)

#Or make a legend as a separate plot:
makePlot.legend.box(plotter);
```

---

makePlot.mapping.rates

Plot mapping rates

Description

Plots the rates at which reads map to the genome.

Usage

```
makePlot.mapping.rates(plotter, plot.mm = NULL,
                    y.counts.in.millions = TRUE,
                    debugMode, singleEndMode,
                    plot = TRUE,
                    ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
plot.mm	Plot multi-mapping rates. By default, it autodetects whether multimapping data was included in the original decoder.
y.counts.in.millions	Label/scale the y-axis in millions of reads.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

For each sample-run, this function plots the number of mapped reads and the rate at which reads map (if the total reads is provided).

See Also

build.plotter read.qc.results.data

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.mapping.rates(plotter);
```

---

makePlot.missingness.rate

Plot N-Rate by read cycle

Description

Plots rate by which the sequencer cannot determine the base, by read cycle.

Usage

```
makePlot.missingness.rate(plotter, r2.buffer = NULL,
                          debugMode, singleEndMode,
                          rasterize.plotting.area = FALSE,
                          raster.height = 1000,
                          raster.width = 1000,
                          log.y = FALSE,
                          ylim = NULL,
                          plot = TRUE,
                          ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
r2.buffer	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
rasterize.plotting.area	Logical. If TRUE, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is FALSE.
raster.height	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the height of the rasterized plot, in pixels.
raster.width	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the width of the rasterized plot, in pixels.
log.y	Logical. If TRUE, the y-axis will be log-scale.
ylim	Numeric of length 2. Sets the y-axis limits.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

x-axis: Read Cycle

y-axis: Rate at which the sequencer assigns an 'N' base.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.missingness.rate(plotter);
```

makePlot.norm.factors
Plot Alignment Clipping

Description

Plots the rate at which the aligner soft-clips off portions of aligned reads.

Usage

```
makePlot.norm.factors(plotter, by.sample = TRUE,
                      return.table = FALSE,
                      debugMode, singleEndMode,
                      plot = TRUE,
                      ...)
makePlot.norm.factors.vs.TC(plotter,
                      by.sample = TRUE,
                      return.table = FALSE,
                      debugMode, singleEndMode,
                      plot = TRUE,
                      ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
by.sample	Logical. Whether to combine all lanebams for each sample before calculating normalization factors. By default, normalization factors for each lanebam will be calculated seperately.
return.table	Logical. Whether to return a data table containing the data that is plotted.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

makePlot.norm.factors plots the normalization factors for each sample/lanebam. Note that unless DESeq2 and edgeR are installed, it will only plot total-count normalization by default. Also note that unless calc.DESeq2 = TRUE and/or calc.edgeR = TRUE in the execution of the read.qc.results.data that produced the QC results, only the total counts normalization factors will be calculated.

makePlot.norm.factors.vs.TC plots the ratio of alternative normalization factors against the total count normalization.

Value

Usually returns nothing, unless return.table is TRUE, in which case it returns a data.frame containing the plotted data for each sample.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.norm.factors(plotter);
makePlot.norm.factors.vs.TC(plotter);

#Legend:
makePlot.legend.box(plotter);
```

---

makePlot.NVC.clip.matchByClipPosition

Plot clipped nucleotide rates, aligned by the distance from the point of clipping.

Description

WARNING: THESE FUNCTIONS ARE BETA, AND ARE NOT FULLY TESTED OR READY FOR GENERAL USE.

Usage

```
makePlot.NVC.lead.clip.matchByClipPosition(plotter,
            clip.amt,  r2.buffer,
            label.majority.bases = TRUE,
            label.majority.bases.threshold = 0.5,
            label.majority.bases.cex = 1,
            rasterize.plotting.area = FALSE,
            raster.height = 1000,
            raster.width = 1000,
            show.base.legend = TRUE,
            load.results = TRUE,
            debugMode, singleEndMode,
            plot = TRUE,
            ...)

makePlot.NVC.tail.clip.matchByClipPosition(plotter,
            clip.amt, r2.buffer,
            label.majority.bases = TRUE,
            label.majority.bases.threshold = 0.5,
            label.majority.bases.cex = 1,
            rasterize.plotting.area = FALSE,
            raster.height = 1000,
```

```
raster.width = 1000,
show.base.legend = TRUE,
debugMode, singleEndMode,
plot = TRUE,
...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
clip.amt	UNDOCUMENTED
r2.buffer	UNDOCUMENTED
label.majority.bases	UNDOCUMENTED
label.majority.bases.threshold	UNDOCUMENTED
label.majority.bases.cex	UNDOCUMENTED
show.base.legend	UNDOCUMENTED
rasterize.plotting.area	Logical. If TRUE, the plotting area will be written to a temp png file then drawn to the current device as a raster image. This option is generally preferred for vector devices, because NVC plots can be very large when drawn in vector format. *Note: this requires the* png *package!*
raster.height	Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.
raster.width	Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels.
load.results	UNDOCUMENTED
debugMode	UNDOCUMENTED
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

This function is currently BETA, and is not intended for general use. Documentation and testing is still pending.

See Also

build.plotter, makePlot.NVC

makePlot.NVC.lead.clip

Clipped sequence nucleotide-by-position plot

Description

Plots the nucleotide rates for clipped segments

Usage

```
makePlot.NVC.lead.clip(plotter, clip.amt,  r2.buffer,
                    points.highlighted = TRUE,
                    label.majority.bases = TRUE,
                    label.majority.bases.threshold = 0.5,
                    label.majority.bases.cex = 1,
                    rasterize.plotting.area = FALSE,
                    raster.height = 1000,
                    raster.width = 1000,
                    show.base.legend = TRUE,
                    debugMode, singleEndMode,
                    plot = TRUE,
                    ...)


makePlot.NVC.tail.clip(plotter, clip.amt,  r2.buffer,
                        points.highlighted = TRUE,
                        label.majority.bases = TRUE,
                        label.majority.bases.threshold = 0.5,
                        label.majority.bases.cex = 1,
                        rasterize.plotting.area = FALSE,
                        raster.height = 1000,
                        raster.width = 1000,
                        show.base.legend = TRUE,
                        debugMode, singleEndMode,
                        plot = TRUE,
                        ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
clip.amt	Numeric value. The number of bases clipped. These methods only plot the sequence for a single specific clip.amt.
r2.buffer	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
points.highlighted	A logical value. Determines whether to ever plot points on top of the lines. This can be useful for identifying outliers. If TRUE, then all highlighted lane-bams will be overlaid with points using their designated pch symbol. If the plotter does not highlight any lanebams, then points will be overlaid on ALL lines.

label.majority.bases
: A logical value indicating whether to label the genotypes of read cycles in which the most common base has a frequency exceeding label.majority.bases.threshold (see below).

label.majority.bases.threshold
: A numeric value indicating the cutoff above which the most common base will be labelled, if label.majority.bases is set TRUE (see above).

label.majority.bases.cex
: The cex value fed to text() that is used to determine how big the labels are to be, if label.majority.bases is TRUE. (see par for information on cex).

rasterize.plotting.area
: Logical. If TRUE, the plotting area will be written to a temp png file then drawn to the current device as a raster image. This option is generally preferred for vector devices, because NVC plots can be very large when drawn in vector format. *Note: this requires the* png *package!*

raster.height
: Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.

raster.width
: Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels.

show.base.legend
: A logical value indicating whether to print the base-color legend along the right edge of the plot.

debugMode
: Logical. If TRUE, debugging data will be printed to the console.

singleEndMode
: Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

plot
: Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

...
: Arguments to be passed to methods, such as graphical parameters (see par).

Details

For general information on reading NVC plots, see makePlot.NVC.

This function plots the nucleotide rates for the sections of reads that were soft-clipped by the aligner. Note that these will not function on reads that have been aligned using an aligner that does not practice soft clipping (such as TopHat2).

See Also

build.plotter, makePlot.NVC

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.NVC.lead.clip(plotter, clip.amt = 12);
makePlot.NVC.tail.clip(plotter, clip.amt = 12);
```

`makePlot.onTarget` *On-Target Rates*

Description

On-Target Rates.

Usage

```
makePlot.targetDistribution(plotter,
       plot.rates = TRUE,
       plot.hist = TRUE,
       log.y = TRUE,
       singleEndMode,
       byPair = !singleEndMode,
       rasterize.plotting.area = FALSE,
       raster.height = 1000,
       raster.width = 1000,
       debugMode,
       plot = TRUE,
       ...)

makePlot.onTarget.counts(plotter,
      y.counts.in.millions = TRUE,
      debugMode,
      singleEndMode,
      plot = TRUE, ...)

makePlot.onTarget.rates(plotter,
      debugMode,
      singleEndMode,
      plot = TRUE, ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`y.counts.in.millions`	Logical. If TRUE, y axis labels are labelled in millions.
`plot.rates`	Logical. If TRUE, plot rates, otherwise plot raw counts.
`plot.hist`	Plot a histogram.
`log.y`	Logical. If TRUE, the y-axis should be log-scaled.
`byPair`	Logical. If TRUE, count read-pairs rather than reads. In other words, if two reads overlap, the overlapping region is counted only once.
`rasterize.plotting.area`	Logical. If `TRUE`, the plotting area will be written to a temp png file then drawn to the current device as a raster image. This option is generally preferred for vector devices, because NVC plots can be very large when drawn in vector format. *Note: this requires the* `png` *package!*

raster.height
: Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.

raster.width
: Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels.

debugMode
: Logical. If TRUE, debugging data will be printed to the console.

singleEndMode
: Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

plot
: Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

...
: Arguments to be passed to methods, such as graphical parameters (see par).

Details

For each sample run, indicates the amount of time spent running the QoRTs QC data processing tool

Value

By default, this function returns nothing. If the return.table parameter is TRUE, then it returns a data.frame with the data that was plotted.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.runTimePerformance(plotter);
```

---

makePlot.overlap *Overlap Mismatch Rates*

Description

For paired-end data only, plots the rate at which the two reads have point-mismatch on overlapping regions.

Usage

```
makePlot.overlap.coverage(plotter,
                          plot.rates = TRUE,
                          singleEndMode,
                          rasterize.plotting.area = FALSE,
                          raster.height = 1000,
                          raster.width = 1000,
```

```
                                debugMode,
                                r2.buffer=NULL,
                                plot = TRUE,
                                ...)

makePlot.overlapMismatch.size(plotter,
                                  plot.rates = TRUE,
                                  log.y = TRUE,
                                  noIndel = TRUE,
                                  draw.decade.lines = c(TRUE,TRUE),
                                  xlim = NULL,
                                  pct.cutoff=0.95,
                                  singleEndMode,
                                  debugMode,
                                  rasterize.plotting.area = FALSE,
                                  raster.height = 1000,
                                  raster.width = 1000,
                                  plot = TRUE,
                                  ...)

makePlot.overlapMismatch.byQual.min(plotter,
                                        plot.rates = TRUE,
                                        log.y = TRUE,
                                        noIndel = TRUE,
                                        draw.decade.lines = TRUE,
                                        singleEndMode,
                                        debugMode,
                                        rasterize.plotting.area = FALSE,
                                        raster.height = 1000,
                                        raster.width = 1000,
                                        plot = TRUE,
                                        ...)

makePlot.overlapMismatch.byQual.read(plotter,
                                         plot.rates = TRUE,
                                         noIndel = TRUE,
                                         log.y = TRUE,
                                         draw.decade.lines = TRUE,
                                         singleEndMode,
                                         debugMode,
                                         rasterize.plotting.area = FALSE,
                                         raster.height = 1000,
                                         raster.width = 1000,
                                         r2.buffer = NULL,
                                         plot = TRUE,
                                         ...)

makePlot.overlapMismatch.byBase(plotter,
                                    noIndel = TRUE,
                                    plot.rates = TRUE,
                                    log.y = FALSE,
                                    y.rate.per.kb = FALSE,
```

```
                                        debugMode,
                                        draw.vert.dotted.lines = TRUE,
                                        singleEndMode,
                                        plot = TRUE,
                                        ...)

  makePlot.overlapMismatch.byBase.atScore(plotter,
                                          atScore = 41,
                                          overlapScoreMethod = c("pair","min"),
                                          plot.rates = TRUE,
                                          noIndel = TRUE,
                                          log.y = FALSE,
                                          singleEndMode,
                                          rasterize.plotting.area = FALSE,
                                          raster.height = 1000,
                                          raster.width = 1000,
                                          debugMode = DEFAULTDEBUGMODE,
                                          plot = TRUE,
                                          ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`atScore`	Numeric integer. For makePlot.overlapMismatch.byBase.atScore, plot the overlap mismatch rates for each base-pair swap at the given phred score. If `overlapScoreMethod` is "min", then this will plot the rate of overlap mismatches where the minimum of the two reads' quality score is `atScore`. If `overlapScoreMethod` is "pair", then this will plot the rate of overlap mismatches where both reads have the same quality score, equal to `atScore`.
`overlapScoreMethod`	Character string. See documentation for atScore, above.
`plot.rates`	A logical value indicating whether or not to plot mismatch rates or mismatch counts.
`log.y`	Logical. If TRUE, the y-axis will be log-scaled.
`noIndel`	Logical. If TRUE, only count overlapping reads if both reads have no aligned indels.
`draw.decade.lines`	Logical. If TRUE, draw mini tick lines at decade points on the y-axis when plotting in log-scale.
`xlim`	Numeric. The x-axis limits. If NULL, the x-limits will be autodetected using the pct.cutoff value.
`pct.cutoff`	Numeric. The percentile cutoff value for the x-axis upper limit.
`debugMode`	Logical. If TRUE, debugging data will be printed to the console.
`singleEndMode`	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
`rasterize.plotting.area`	Logical. If `TRUE`, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package.

Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is FALSE.

raster.height
: Numeric integer. If rasterize.plotting.area is TRUE, then this will set the height of the rasterized plot, in pixels.

raster.width
: Numeric integer. If rasterize.plotting.area is TRUE, then this will set the width of the rasterized plot, in pixels.

r2.buffer
: Numeric. The desired distance between the read1/read2 sub-plots.

plot
: Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

y.rate.per.kb
: Logical. If TRUE, the y axis should be labelled in overlap rate per kilobase.

draw.vert.dotted.lines
: Logical. If TRUE, draw dotted lines at decade y axis points.

...
: Arguments to be passed to methods, such as graphical parameters (see par).

Details

These plotting functions create plots that summarize the overlap-mismatch between paired end reads. When paired-end reads overlap, the sequence on the two reads can be compared in order to estimate the sequencer error rate. This error rate can be visualized in a variety of ways to detect sequencer artifacts or errors.

makePlot.overlap.coverage: The rates at which each position in read 1 and read 2 are covered by the other read. x-axis: Read position. y-axis: How often the given position on the given read is covered by the other read.

makePlot.overlap.mismatchCountPerRead: A histogram of the amount of mismatch between the two reads. x-axis: Number of mismatches. y-axis: How often a read is observed with the given number of mismatches.

makePlot.overlap.mismatch.byMinQual: The mismatch rate as a function of the minimum of the two base quality scores. x-axis: Minimum quality score y-axis: Mismatch rate.

makePlot.overlap.mismatch.byAvgQual: The mismatch rate as a function of the average of the two base quality scores. x-axis: Average quality score y-axis: Mismatch rate.

makePlot.overlap.mismatch.byReadQual: The mismatch rate as a function of each read's quality score. x-axis: Quality score for read 1 and read 2. y-axis: Mismatch rate.

makePlot.overlap.mismatch.byBase: The rate at which overlap-mismatches occur for each possible base-pair swap. x-axis: Read1/read2 base-pair values (note: read2 base is complemented). y-axis: Mismatch rate.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.overlap.coverage(plotter)
makePlot.overlapMismatch.size(plotter)
makePlot.overlapMismatch.byQual.min(plotter)
```

```
makePlot.overlapMismatch.byQual.read(plotter)
makePlot.overlapMismatch.byBase(plotter)
makePlot.overlapMismatch.byBase.atScore(plotter)
```

`makePlot.qual.pair` *Plot quality score by read cycle*

Description

Plots the Phred quality score as a function of the read cycle for both reads.

Usage

```
makePlot.qual.pair(plotter, y.name, r2.buffer = NULL,
                debugMode, singleEndMode,
                rasterize.plotting.area = FALSE, raster.height = 1000, raster.w
                plot = TRUE,
                ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`y.name`	The name of the quality score metric to plot. Must be one of: • `"min"` • `"lowerQuartile"` • `"median"` • `"upperQuartile"` • `"max"`
`r2.buffer`	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
`debugMode`	Logical. If TRUE, debugging data will be printed to the console.
`singleEndMode`	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
`rasterize.plotting.area`	Logical. If `TRUE`, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is `FALSE`.
`raster.height`	Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the height of the rasterized plot, in pixels.
`raster.width`	Numeric integer. If `rasterize.plotting.area` is `TRUE`, then this will set the width of the rasterized plot, in pixels.
`plot`	Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

These plots display information about the phred quality score (y-axis) as a function of the position in the read (x-axis). Five statistics can be plotted: minimum, maximum, upper and lower quartiles, and median. These statistics are calculated individually for each bam file and each read position (ie, each plotted line corresponds to a bam file).

Note that the Phred score is always an integer, and as such these plots would normally be very difficult to read because lines would be plotted directly on top of one another. To reduce this problem, the lines are vertically offset from one another. Most plotters offset each line by lane.ID.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.qual.pair(plotter,y.name="min");
makePlot.qual.pair(plotter,y.name="median");
makePlot.qual.pair(plotter,y.name="max");

#Legend:
makePlot.legend.box(plotter);
```

---

makePlot.raw.NVC *Nucleotide-rate by read cycle plot*

Description

Plots the nucleotide rate by read cycle

Usage

```
makePlot.raw.NVC(plotter,  r2.buffer = NULL,
                         points.highlighted = TRUE,
                         label.majority.bases = FALSE,
                         label.majority.bases.threshold = 0.5,
                         label.majority.bases.cex = 0.5,
                         rasterize.plotting.area = FALSE,
                         raster.height = 1000,
                         raster.width = 2000,
                         show.base.legend = TRUE,
                         debugMode, singleEndMode,
                         useFQ = FALSE,
                         plot = TRUE,
                         ...)

makePlot.minus.clipping.NVC(plotter,  r2.buffer = NULL,
                         points.highlighted = TRUE,
                         label.majority.bases = FALSE,
                         label.majority.bases.threshold = 0.5,
```

```
                        label.majority.bases.cex = 0.5,
                        rasterize.plotting.area = FALSE,
                        raster.height = 1000,
                        raster.width = 2000,
                        show.base.legend = TRUE,
                        debugMode, singleEndMode,
                        plot = TRUE,
                        ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
r2.buffer	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
points.highlighted	A logical value. Determines whether to ever plot points on top of the lines. This can be useful for identifying outliers. If TRUE, then all highlighted lane-bams will be overlaid with points using their designated pch symbol. If the plotter does not highlight any lanebams, then points will be overlaid on ALL lines.
label.majority.bases	A logical value indicating whether to label the genotypes of read cycles in which the most common base has a frequency exceeding label.majority.bases.threshold (see below).
label.majority.bases.threshold	A numeric value indicating the cutoff above which the most common base will be labelled, if label.majority.bases is set TRUE (see above).
label.majority.bases.cex	The cex value fed to text() that is used to determine how big the labels are to be, if label.majority.bases is TRUE. (see par for information on cex).
rasterize.plotting.area	Logical. If TRUE, the plotting area will be written to a temp png file then drawn to the current device as a raster image. This option is generally preferred for vector devices, because NVC plots can be very large when drawn in vector format. *Note: this requires the* png *package!*
raster.height	Numeric. If rasterize.plotting.area is TRUE, then this is the height of the plotting area raster image, in pixels.
raster.width	Numeric. If rasterize.plotting.area is TRUE, then this is the width of the plotting area raster image, in pixels.
show.base.legend	A logical value indicating whether to print the base-color legend along the right edge of the plot.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
useFQ	Logical. If TRUE, plot the G/C rate from the unaligned FASTQ data. This requires that the FASTQ data was supplied to QoRTs in the QoRTs QC step.

plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

This plot displays the nucleotide rates by read cycle. The color scheme for NVC plots is different from all other plots. Rather than being used for emphasis or to allow cross-comparisons by sample, biological-condition, or lane, the colors are used to indicate the four nucleotides: A (green), T (red), G (orange), or C (blue). Depending on the type of QoRT_Plotter being used, sample-runs will be marked and differentiated by marking the lines with shapes (R points). In many cases the points will be unreadable due to overplotting, but clear outliers that stray from the general trends can be readily identified.

When used with a "sample.highlight" type QoRT_Plotter (see build.plotter), "highlighted" samples will be drawn with a deeper shade of the given color.

makePlot.raw.NVC plots the nucleotides of all cycles of all aligned reads. makePlot.minus.clipping.NVC plots the nucleotides for all cycles that are NOT soft-clipped by the aligner. Note that for reads aligned with an aligner that does not perform soft-clipping (such as TopHat2), the two plots will be identical.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.minus.clipping.NVC(plotter);
makePlot.raw.NVC(plotter);
```

---

makePlot.readLengthDist

Plot the distribution of read lengths.

Description

Plots the distribution of read lengths. Only useful for data with variable trimming (which is generally not recommended at least for RNA-Seq data).

Usage

```
makePlot.readLengthDist(plotter,
                        plot.rates = TRUE,
                        plot.means = TRUE,
                        plot.medians = NULL,
                        include.full.length = FALSE,
                        cumulative = TRUE,
                        singleEndMode,
                        rasterize.plotting.area = FALSE,
```

```
                                raster.height = 1000,
                                raster.width = 1000,
                                debugMode,
                                r2.buffer,
                                plot = TRUE,
                                ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
plot.rates	A logical value indicating whether or not the X-axis should be the raw number of nucleotides that are G/C, vs the rate G/C.
plot.means	A logical value indicating whether or not to plot the mean average GC content for each bam file at the bottom of the plot.
plot.medians	A logical value indicating whether or not to plot the median average GC content for each bam file at the bottom of the plot. Overrides plot.means.
include.full.length	Logical. If FALSE, omit the full-length read length from the x-axis of the plot.
cumulative	Logical. If TRUE, plot shows cumulative rates.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
rasterize.plotting.area	Logical. If TRUE, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is FALSE.
raster.height	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the height of the rasterized plot, in pixels.
raster.width	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the width of the rasterized plot, in pixels.
r2.buffer	Buffer space to place between the plotting of read 1 and read 2. By default this will choose a reasonable value.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

x-axis: Read Length

y-axis: Percentage of reads with length equal to the given length. If cumulative == TRUE, then it is the percentage of reads with length less than or equal to the given length.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.readLengthDist(plotter)
```

---

makePlot.reference *Plot Reference Mismatch Rates*

Description

Plots various rates of single-base mismatches against the reference genome.

Usage

```
makePlot.referenceMismatch.byScore(plotter,
             plot.rates = TRUE,
             draw.decade.lines = c(FALSE,TRUE),
             log.y = TRUE,
             singleEndMode = plotter$res@singleEnd,
             debugMode = DEFAULTDEBUGMODE,
             rasterize.plotting.area = FALSE,
             raster.height = 1000,
             raster.width = 1000,
             r2.buffer = NULL,
             plot = TRUE,
             ...)

makePlot.referenceMismatch.byBase(plotter,
             y.rate.per.kb = TRUE, draw.vert.dotted.lines = TRUE,
             debugMode = DEFAULTDEBUGMODE,
             singleEndMode = plotter$res@singleEnd,
             plot = TRUE,
             ...)

makePlot.referenceMismatch.byBase.atScore(plotter,
             atScore = 41,
             forRead = c("BOTH","R1","R2"),
             plot.rates = TRUE,
             log.y = FALSE,
             singleEndMode = plotter$res@singleEnd,
             rasterize.plotting.area = FALSE,
             raster.height = 1000,
             raster.width = 1000,
             debugMode = DEFAULTDEBUGMODE,
             plot = TRUE,...)

makePlot.referenceMismatch.byCycle(plotter,
             plot.rates = TRUE,
             log.y = TRUE,
             ylim = NULL,
             singleEndMode = plotter$res@singleEnd,
```

```
                debugMode = DEFAULTDEBUGMODE,
                rasterize.plotting.area = FALSE,
                raster.height = 1000,
                raster.width = 1000,
                r2.buffer = NULL,
                plot = TRUE,
                ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
plot.rates	A logical value indicating whether or not to plot mismatch rates or mismatch counts.
log.y	Logical. If TRUE, the y-axis will be log-scaled.
draw.decade.lines	Logical. If TRUE, draw mini tick lines at decade points on the y-axis when plotting in log-scale.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
rasterize.plotting.area	Logical. If TRUE, then "flatten" the plotting lines into a raster format. This requires support for png file creation and the installation of the "png" package. Only the plotting lines will be rasterized, the axes and annotations will be vector format. Default is FALSE.
raster.height	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the height of the rasterized plot, in pixels.
raster.width	Numeric integer. If rasterize.plotting.area is TRUE, then this will set the width of the rasterized plot, in pixels.
r2.buffer	Numeric. The desired distance between the read1/read2 sub-plots.
y.rate.per.kb	Logical. If TRUE, then y-axis should be labelled as the rate per kilobase.
draw.vert.dotted.lines	Logical. If TRUE, then draw dotted guide-lines at reasonable intervals.
atScore	Numeric integer. For makePlot.referenceMismatch.byBase.atScore, this sets the PHRED score.
forRead	Character string. For makePlot.referenceMismatch.byBase.atScore, this sets whether the plot should be for read 1, read 2, or both side by side.
ylim	Numeric vector of length 2. This sets the y-axis limits.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

These plotting functions create plots that summarize the rate of single-base-pair reference-mismatches of reads against the reference genome. A "reference mismatch" is defined as an aligned read base that does not match the genomic base.

These reference mismatches may be caused by a number of different factors and may not necessarily indicate a serious quality issue. To begin with, many reference mismatches may be caused by real single nucleotide polymorphisms present in the subject. QoRTs produces several different plots that allow these mismatch rates to be measured in a variety of ways.

`makePlot.referenceMismatch.byCycle`: Plots the overall reference mismatch rate as a function of read cycle.

`makePlot.referenceMismatch.byBase`: Plots the overall reference mismatch rate for each of the 12 possible X -> Y base-pair substitutions.

`makePlot.referenceMismatch.byScore`: Plots the reference mismatch rate as a function of the PHRED score.

`makePlot.referenceMismatch.byBase.atScore`: Plots the reference mismatch rate for each of the 12 possible X -> Y base-pair substitutions, for the subset of bases with PHRED score equal to `atScore`.

NOTE: Creation of the reference mismatch plots requires that the QoRTs QC run be executed with additional optional parameters. The BAM file must be sorted by coordinate, and a genome fasta file must be specified via the `--genomeFA` parameter.

See Also

`build.plotter`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
```

---

makePlot.runTimePerformance

Chromosome type rate plot

Description

Plots the number or percent of read-pairs falling on each type of chromosome.

Usage

```
makePlot.runTimePerformance(plotter,
                debugMode, singleEndMode,
                plot = TRUE,
                ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

For each sample run, indicates the amount of time spent running the QoRTs QC data processing tool

Value

By default, this function returns nothing. If the return.table parameter is TRUE, then it returns a data.frame with the data that was plotted.

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.runTimePerformance(plotter);
```

---

makePlot.splice.junction.event.rates
Plot Splice Junction Event Rates

Description

Plots the rates at which splice junctions occur.

Usage

```
makePlot.splice.junction.event.counts(plotter,
                                      high.low.cutoff = 4,
                                      debugMode, singleEndMode,
                                      plot = TRUE,
                                      ...)
makePlot.splice.junction.event.ratesPerRead(plotter,
                                      high.low.cutoff = 4,
                                      debugMode, singleEndMode,
```

```
                                    plot = TRUE,
                                    ...)
    makePlot.splice.junction.event.proportions(plotter,
                                    high.low.cutoff = 4,
                                    debugMode, singleEndMode,
                                    plot = TRUE,
                                    ...)
    makePlot.splice.junction.event.proportionsByType(plotter,
                                    high.low.cutoff = 4,
                                    debugMode, singleEndMode,
                                    plot = TRUE,
                                    ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`high.low.cutoff`	For Advanced users only! The cutoff between high and low expression splice junctions. Note that in order to function, this same cutoff MUST be used by the QoRTs jar utility that generates these counts.
`debugMode`	Activates debug mode, which causes more verbose reporting.
`singleEndMode`	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
`plot`	Logical. If FALSE, suppress plotting and return `TRUE` if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`).

Details

These functions plot various metrics for the rate at which splice junction "events" occur. A splice junction "event" is an occurrance of a mapped read-pair bridging a splice junction. Some read-pairs may contain multiple splice junction events, and some read-pairs may contain none.

Splice junctions are characterized into six categories:

- Known: The splice junction locus is found in the supplied transcript annotation gtf file.
- Novel: The splice junction locus is NOT found in the supplied transcript annotation gtf file.
- Known, 1-3 reads: The locus is known, and is only covered by 1-3 read-pairs.
- Known, 4+ reads: The locus is known, and is covered by 4 or more read-pairs.
- Novel, 1-3 reads: The locus is novel, and is only covered by 1-3 read-pairs.
- Novel, 4+ reads: The locus is novel, and is covered by 4 or more read-pairs.

`makePlot.splice.junction.event.counts` plots the number (y-axis) of all splice junction events falling into each of six categories. Note that because different samples/runs may have different total read counts and/or library sizes, this function is generally not the best for comparing between samples. For most purposes, `makePlot.splice.junction.event.ratesPerRead` will be preferable.

`makePlot.splice.junction.event.ratesPerRead` plots the rate (y-axis) at which each type of splice junction events appear, per read-pair.

`makePlot.splice.junction.event.proportions` plots the proportion (y-axis) of all splice junction events falling into the six categories.

`makePlot.splice.junction.event.proportionsByType` plots the proportion (y-axis) at which splice junction events appear on known vs novel splice junction loci, the proportion of known splice junction events that occur on low-coverage junctions (1-3 read-pairs) vs high-coverage junctions (4 or more read-pairs), and the proportion of novel splice junction events that occur on low vs high coverage junctions.

All of these plots are generally used to detect whether sample-specific or batch effects have a substantial or biased effect on splice junction appearance, either due to differences in the original RNA, or due to artifacts that alter the rate at which the aligner maps across splice junctions.

See Also

`build.plotter`, `makePlot.splice.junction.loci.counts`

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.splice.junction.event.counts(plotter);
makePlot.splice.junction.event.ratesPerRead(plotter);
makePlot.splice.junction.event.proportions(plotter);
makePlot.splice.junction.event.proportionsByType(plotter);

#Legend:
makePlot.legend.box(plotter);
```

---

makePlot.splice.junction.loci.counts

Splice Junction Loci Count Plot

Description

Plots the rate at which splice junction loci fall into various categories.

Usage

```
makePlot.splice.junction.loci.counts(plotter,
                                     calc.rate = FALSE,
                                     high.low.cutoff = 4,
                                     debugMode, singleEndMode,
                                     plot = TRUE, ...)
```

Arguments

`plotter`	A `QoRT_Plotter` reference object. See `build.plotter`.
`calc.rate`	Logical. If TRUE, the proportion of loci in each category will be calculated and plotted, rather than the raw number.
`high.low.cutoff`	(For advanced users only!) The cutoff between "high" and "low" expression junction loci. Note that this must match the cutoff used by the jarfile QC execution.

debugMode	Logical. If TRUE, debugging data will be printed to the console.
singleEndMode	Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.
plot	Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

This function plots the number (y-axis) of splice junction *loci* of each type that appear in the bam-file's reads. Splice junctions are split into 4 groups, first by whether the splice junction appears in the transcript annotation gtf ("known" vs "novel"), and then by whether the splice junction has 4 or more reads covering it, or 1-3 reads ("Hi" vs "Lo").

See Also

build.plotter, makePlot.splice.junction.event

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.splice.junction.loci.counts(plotter);
#Add a legend:
makePlot.legend.over("topright", plotter)
```

---

makePlot.strandedness.test

Strandedness Test Plot

Description

Plots the apparent strandedness of the reads.

Usage

```
makePlot.strandedness.test(plotter, plot.target.boxes = FALSE,
                           debugMode, singleEndMode, plot = TRUE, ...)
```

Arguments

plotter	A QoRT_Plotter reference object. See build.plotter.
plot.target.boxes	A logical value. If true, then green target boxes will be printed over the area in which all points should be expected to fall.
debugMode	Logical. If TRUE, debugging data will be printed to the console.

singleEndMode
: Logical. Determines whether the dataset consists of single-ended reads. By default this is determined from the data. Thus, this parameter should never need to be set by the user.

plot
: Logical. If FALSE, suppress plotting and return TRUE if and only if the data is present in the dataset to allow plotting. Useful to automatically filter out missing data plots.

...
: Arguments to be passed to methods, such as graphical parameters (see par).

Details

TODO!

See Also

build.plotter

Examples

```
data(res,package="QoRTsExampleData");
plotter <- build.plotter.colorByGroup(res);
makePlot.strandedness.test(plotter);

#Add a legend:
makePlot.legend.over("topright", plotter)
```

---

read.qc.results.data

Reading QC results data

Description

Creates a QoRT_QC_Results object using a set of QC result data files.

Usage

```
read.qc.results.data(infile.dir,
                     decoder,
                     decoder.files,
                     calc.DESeq2 = FALSE,
                     calc.edgeR = FALSE,
                     debugMode,
                     autodetectMissingSamples = FALSE,
                     skip.files = c())

completeAndCheckDecoder(decoder, decoder.files)
```

Arguments

`infile.dir`	REQUIRED. The base file directory where all the QC results data is stored.
`decoder`	A character vector or data.frame containing the decoder information. See details below.
`decoder.files`	Character vector. Either one or two character strings. Either decoder.files OR decoder must be set, never both. See details below.
`calc.DESeq2`	Logical. If TRUE, this function will attempt to load the DESeq2 package. If the DESeq2 package is found, it will then calculate DESeq2's geometric normalization factors (also known as "size factors") for each replicate and for each sample.
`calc.edgeR`	Logical. If TRUE, this function will attempt to load the edgeR package. If the edgeR package is found, it will then calculate all of edgeR's normalization factors (also known as "size factors") for each replicate and for each sample.
`debugMode`	Logical. If TRUE, debugging data will be printed to the console.
`autodetectMissingSamples`	Logical. If TRUE, automatically drop replicates for which the QC files cannot be found. By default this function will throw an error.
`skip.files`	Character vector of QC data file titles to skip. This can be useful for performing a faster data load when only querying a subset of the available QC metrics. See examples below.

Details

read.qc.results.data reads in a full QoRTs dataset of multiple QoRTs QC runs and compiles them into a QoRTs_Results object.

completeAndCheckDecoder simply reads a decoder and "fills in" all missing parameters, returning a data.frame.

The "decoder" is used to describe each replicate/sample. The standard decoder is a data frame that has one row per replicate, with the following columns:

- unique.ID: The base identifier for the individual replicate. Must be unique. lanebam.ID is a synonym.
- lane.ID: (OPTIONAL) The identifier for the lane, run, or batch. The default is "UNKNOWN".
- group.ID: (OPTIONAL) The identifier for the biological condition for the given replicate. The default is "UNKNOWN".
- sample.ID: (OPTIONAL) The identifier for the specific biological replicate from which the replicate belongs. Note that this is distinct from "lanebam.ID" because in many RNA-Seq studies each "sample" can have multiple technical replicates, as multiple sequencing runs may be needed to acquire sufficient reads for analysis. By default, it is assumed that each replicate comes from a different sample, and sample.ID is set to equal unique.ID.
- qc.data.dir: (OPTIONAL) This column indicates the subdirectory in which the replicate's QC data was written. If this column is missing, it is assumed to be equal to the unique.ID.
- input.read.pair.count: (OPTIONAL) This column contains the number of read-pairs (or just reads, for single-end data), before alignment. This is used later to calculate mapping rate.
- multi.mapped.read.pair.count: (OPTIONAL) This column contains the number of read-pairs (or just reads, for single-end data) that were multi-mapped. This must be included for multi-mapping rate to be calculated.

All the parameters except for unique.ID are optional. The decoder can even be supplied as a simple character vector, which is assumed to be the unique.ID's. All the other variables will be set to their default values.

Alternatively, the decoder can be supplied as a file given by the decoder.files parameter.

Dual Decoder: Optionally, two decoders can be supplied. In this case the first decoder should be the technical-replicate decoder and the second should be the biological-replicate decoder. The technical-replicate decoder should have one row per unique.ID, with the following columns:

- unique.ID: The base identifier for the individual replicate. Must be unique!
- lane.ID: (OPTIONAL) The identifier for the lane, run, or batch.
- sample.ID: (OPTIONAL) The identifier for the specific biological replicate from which the replicate belongs. Note that this is distinct from "lanebam.ID" because in many RNA-Seq studies each "sample" can have multiple technical replicates, as multiple sequencing runs may be needed to acquire sufficient reads for analysis.
- qc.data.dir: (OPTIONAL) This column indicates the subdirectory in which the replicate's QC data was written. If this column is missing, it is assumed to be equal to the lanebam.ID. Must be unique!
- input.read.pair.count: (OPTIONAL) This column contains the number of input reads, before alignment. This is used later to calculate mapping rate.
- multi.mapped.read.pair.count: (OPTIONAL) This column contains the number of reads that were multi-mapped. This must be included for multi-mapping rate to be calculated.

The biological-replicate decoder should have one row per sample.ID, with the following columns:

- sample.ID: The identifier for the specific biological replicate from which the replicate belongs. Note that this is distinct from "unique.ID" because in many RNA-Seq studies each "sample" can have multiple technical replicates, as multiple sequencing runs may be needed to acquire sufficient reads for analysis.
- group.ID (OPTIONAL): The identifier for the biological condition for the given replicate.

All decoders are allowed to contain other columns in addition to the ones listed here, so long as their names are distinct. Columns do not need to appear in any particular order, so long as they are named according to the specifications above.

See Also

build.plotter

Examples

```
#Load the decoder from the example dataset:
directory <- paste0(system.file("extdata/",
                    package="QoRTsExampleData",
                    mustWork=TRUE),"/");
decoder.file <- system.file("extdata/decoder.txt",
                            package="QoRTsExampleData",
                            mustWork=TRUE);
decoder.data <- read.table(decoder.file,
                           header=TRUE,
                           stringsAsFactors=FALSE);
print(decoder.data);

#This command produces the example dataset used in all the other
```

```
#  examples:
res <- read.qc.results.data(directory,
                            decoder = decoder.data,
                            calc.DESeq2 = TRUE,
                            calc.edgeR = TRUE);
#Note that DESeq2 and edgeR are required in order to
#  calculate the size factors using the options above.

#You can also specify incomplete decoders, and use
#  the following command to fill in the defaults:
completeAndCheckDecoder(c("SAMP1","SAMP2",
                          "SAMP3","SAMP4",
                          "SAMP5","SAMP6"))

#You don't actually have to use completeAndCheckDecoder,
#You can just pass the incomplete decoder directly to QoRTs.
#For example, to load a small subset of the example data
#(without phenotype data):
res <- read.qc.results.data(paste0(directory,"/ex/"),
                            decoder = c("SAMP1_RG1","SAMP2_RG1",
                                        "SAMP3_RG1","SAMP4_RG1"));

#The list of available QC names for use with skip.files:
names(res@qc.data);
#Skip some of the files using a command like this:
res.quick <- read.qc.results.data(directory,
                            decoder = decoder.data,
                            skip.files=c(
                            "QC.NVC.raw.R1.txt.gz",
                            "QC.NVC.raw.R2.txt.gz",
                            "QC.NVC.lead.clip.R1.txt.gz",
                            "QC.NVC.lead.clip.R2.txt.gz",
                            "QC.NVC.tail.clip.R1.txt.gz",
                            "QC.NVC.tail.clip.R2.txt.gz"
                            ));
```

Index