ggbio

March 24, 2012

`geom_chevron` *Chevron Geoms for GRanges object*

Description

Break normal intervals stroed in `GRanges` object and show them as chevron geom, useful for showing model or splice summary.

Usage

```
geom_chevron(data, ..., group.name, offset = 0.1,
             chevron.height = c(0.1, 0.8))
```

Arguments

`data`	A GRanges object.
`...`	Extra parameters passed to qplot function.
`group.name`	group.name passed to `addSteppings`, in case your `GRanges` object doesn't contain ".levels" column. You can do it here.
`offset`	A nunmeric value or characters. If it's numeric value, indicate how much you want the chevron to wiggle, usually the rectangle for drawing `GRanges` is of height unit 1, so it's better between -0.5 and 0.5 to make it nice looking. Unless you specify offset as one of those columns, this will use height of the chevron to indicate the columns. Of course you could use size of the chevron to indicate the column variable easily, please see the examples.
`chevron.height`	A numeric vector of length 2. When the offset parameters is a character which is one of the data columns, this parameter could specify the wiggled range of the chevron(or the height range).

Details

To draw a normal GRanges as Chevron instead of segments or rectangle as shown in previous sections, we need to provide a special geom for this purpose. Chevron is popular in gene viewer or genomoe browser, when they try to show isoforms or gene model.`geom_chevron`, just like any other `geom_*` function in ggplot2, you can pass aes() to it to use height of chevron or width of chevron to show statistics summary.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
## Not run:
library(GenomicRanges)
gr <- GRanges("chr1", IRanges(c(100, 200, 300), width = 50))
p <- qplot(gr)
gr.gaps <- gaps(gr)[-1]
values(gr.gaps)$score <- c(1, 100)
p + geom_chevron(gr.gaps)
p + geom_chevron(gr.gaps, aes(size = score), offset = "score")
p + geom_chevron(gr.gaps, aes(size = score), offset = "score",
                 chevron.height = c(0.1, 0.2))
p + geom_chevron(gr.gaps, offset = -0.1)

## End(Not run)
```

---

geom_hotregion *Adding hotregion for stacked overview (genome-wide)*

Description

Adding hotregion which is a `GRanges` object for stacked overview (genome-wide)

Usage

```
geom_hotregion(data, ...)
```

Arguments

`data`	A `GRanges` object, which you want to overlay on the stacked overview.
`...`	Extra parameters passed to geom in ggplot2. e.g. aes(color = score)

Details

The overplayed region may contain single position which is not interval, this will be plotted as segments instead of rectangle.

Value

A 'Layer'

Author(s)

Tengfei Yin

Examples

```
data(hg19IdeogramCyto)
library(GenomicRanges)
## make shorter and clean labels
old.chrs <- seqnames(seqinfo(hg19IdeogramCyto))
new.chrs <- gsub("chr", "", old.chrs)
names(new.chrs) <- old.chrs
new.ideo <- renameSeqlevels(hg19IdeogramCyto, new.chrs)
p <- plotOverview(new.ideo, cytoband = FALSE)
data(darned_hg19_subset500)
## rename
new.darned <- renameSeqlevels(darned_hg19_subset500, new.chrs)
p <- p + geom_hotregion(new.darned)
print(p)
```

---

plotFragLength *Plot estimated fragment length for paired-end RNA-seq data*

Description

Plot estimated fragment length for paired-end RNA-seq data against single reduced data model.

Usage

```
## S4 method for signature 'character,GRanges'
plotFragLength(data, model,
               gap.ratio = 0.0025,
               geom = c("segment", "point", "line"),
               type = c("normal", "cut"),
               heights = c(400, 100),
               annotation = TRUE)
```

Arguments

data	A character indicate the bam file.
model	A reduced model to compute estimated fragment length. please see details.
gap.ratio	When type is set to "cut", it will provide a compact view, which cut the common gaps in a certain ratio.
geom	One or all three geoms could be drawn at the same time. y value of "point" and "line" indicate the estimated fragment length. and if geom is set to "segment", the segment is from the left most position to paired right most position, should be equal to "isize".
type	"normal" return a uncut view, loose but the coordinate is true genomic coordinates. "cut" cut the view in a compact way.
heights	Numeric vector indicate the heights of tracks.
annotation	A logical value. TRUE shows model, and FALSE shows only fragment length with labels.

Details

We use a easy way to define this estimated fragment length, we collect all paired reads and model, reduce model first, then find common gaps, remove common gaps between paired-end reads, and compute the new estimated fragment length.

Value

A ggplot object when `annotation = FALSE` and a frame grob if `annotation = TRUE`

Author(s)

Tengfei Yin

Examples

```
## Not run:
data(genesymbol)
bamfile <- system.file("extdata", "SRR027894subRBM17.bam", package="biovizBase")
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- Hsapiens_UCSC_hg19_knownGene_TxDb
model <- exonsBy(txdb, by = "tx")
model.new <- subsetByOverlaps(model, genesymbol["RBM17"])
exons.rbm17 <- subsetByOverlaps(exons(txdb), genesymbol["RBM17"])
exons.new <- reduce(exons.rbm17)
plotFragLength(bamfile, exons.new, geom = "line")
plotFragLength(bamfile, exons.new, geom = c("point","segment"))
plotFragLength(bamfile, exons.new, geom = c("point","segment"), annotation = FALSE)
plotFragLength(bamfile, exons.new, geom = c("point","segment"), type = "cut",
               gap.ratio = 0.001)

## End(Not run)
```

---

plotGrandLinear *Manhattan for GWAS*

Description

A Manhattan plot is special scatter plot used to visualize data with a large number of data points, with a distribute of some higher-magnitude values. For example, in the GWAS(genome-wide association studies). Here we mainly focus on GWAS Manhattan plots. X-axis is genomic coordinates and Y-axis is negative logarithm of the associated P-value for each single nucleotide polymorphism. So higher the value, more stronger the association they are.

Usage

```
plotGrandLinear(obj, y, title, facet,
                size, shape, color, alpha,
                ...,
                geom = c("point", "line"),
                color.type = c("twocolor", "identity", "seqnames"),
                two.color = c("#0080FF", "#4CC4FF"),
                cutoff = NULL,
```

```
  cutoff.color = "red",
  cutoff.size = 1,
  legend = FALSE,
  xlab = "Chromosome",
  ylab = substitute(y),
  theme_bw = TRUE)
```

Arguments

obj	GRanges object which contains extra p value, before users pass this object, they need to make sure the pvalue has been changed to -log10(p).
y	Unevaluated name which should be one the of the column names indicating the p-value Or other score used as y value in the plot. This is required field.
title	Title for the plot, default plot without title.
facet	A facet formula, when provided, should be in two forms. First one, `facet = group ~ seqnames`, where group should be the name you want to facet by rows, if not, use `.` instead. this is faceted by seqnames, the different chromosomes will be in different panel. The second form, `facet = group ~ .` will give you a compact grand linear view, seqnames will not be in different panels, this is a better way to plot a overview, e.g. a Manhattan plot.
size	Size for point or lines. Could equal a column name or a fixed number. When it's fixed, please use `I()` to wrap the value.
shape	Shape for point or lines. Could equal a column name or a fixed number. When it's fixed, please use `I()` to wrap the value.
color	Color for point for lines. Could equal a column name or a fixed character. When it's fixed, please use `I()` to wrap the value.
alpha	Alpha blending. Could equal a column name or a fixed number. When it's fixed, please use `I()` to wrap the value.
...	Extra arguments passed. Will be automatically dispatched to the right place, such as scales, space. When you try to use a faceted seqnames, you need to specify `scales = "free"` and `space = "free"`, if you want a free scaled x-scale.
geom	"point"(default) or "line". When it's line, it will make sure the line is not connect cross different chromosomes.
color.type	"identity" use single color for all points, when it's enabled, you can specify color in the arguments to equal a character or an unevaluated name, when use specific color, try use `I`, for instance, color = I("red"); "seqnames" use default discrete color scheme for all chromosomes; "twocolor" use two colors to represent all the chromosomes, could specify color in the two.color argument.
two.color	A character vector of two. Default is chosen from dichromat palette "BluetoOrange.8", make sure it's color-blind safe.
cutoff	A numeric vector which used as cutoff for Manhattan plot.
cutoff.color	A character specifying the color used for cutoff. Default is "red".
cutoff.size	A numeric value which used as cutoff line size.
legend	A logical value indicate whether to show legend or not. Default is FALSE which disabled the legend.
xlab	Label for xscale.
ylab	Label for yscale.

theme_bw A logical value indicate whether to show gray background or not. Default is TRUE, use theme_bw() in ggplot2.

Details

If scales and space are free, then the mapping between position and values in the data will be the same across all panels

Value

Return a ggplot object.

Author(s)

Tengfei Yin

Examples

```
## Not run:
library(GenomicRanges)
library(ggbio)
data(hg19IdeogramCyto)
data(hg19Ideogram)
chrs <- as.character(levels(seqnames(hg19IdeogramCyto)))
seqlths <- seqlengths(hg19Ideogram)[chrs]
set.seed(1)
nchr <- length(chrs)
nsnps <- 1000
gr.snp <- GRanges(rep(chrs,each=nsnps),
                  IRanges(start = do.call(c, lapply(chrs, function(chr){
                    N <- seqlths[chr]
                    runif(nsnps,1,N)
                  })), width = 1),
                  SNP=sapply(1:(nchr*nsnps), function(x) paste("rs",x,sep='')),
                  pvalue =  -log10(runif(nchr*nsnps)),
                  group = sample(c("Normal", "Tumor"), size = nchr*nsnps,
                    replace = TRUE)
                  )
## processing the name
nms <- seqnames(seqinfo(gr.snp))
nms.new <- gsub("chr", "", nms)
names(nms.new) <- nms
gr.snp <- renameSeqlevels(gr.snp, nms.new)


## compact view
## no facet by samples, but make sure you want it that way
## default is two color
plotGrandLinear(gr.snp, y = pvalue, geom = "point")
## facet by samples, comparison across groups
plotGrandLinear(gr.snp, y = pvalue, geom = "point",
                facet = group ~ .,  color.type = "twocolor")
## change two color
plotGrandLinear(gr.snp, y = pvalue, geom = "point",
                facet = group ~ .,  color.type = "twocolor",
                two.color = c("red", "blue"))
## geom line
```

```
plotGrandLinear(gr.snp, y = pvalue,
                geom = "line", facet = group ~ .)


## add size and change color
plotGrandLinear(gr.snp, y = pvalue, size = pvalue,geom = "point",
                facet = group ~ .,  color.type = "seqnames")

plotGrandLinear(gr.snp, y = pvalue, size = I(0.05),
                geom = "point", facet = group ~ .)

plotGrandLinear(gr.snp, y = pvalue, color = group, geom = "point",
                facet = group ~ .,
                color.type = "identity")

plotGrandLinear(gr.snp, y = pvalue, color = I("blue"), geom = "point", facet = group ~ .,
                color.type = "identity")

## facet by seqnames, slower
plotGrandLinear(gr.snp, y = pvalue,geom = "point",
                facet = group ~ seqnames, scales = "free", space = "free")


## End(Not run)
```

`plotMismatchSum` *Plot mismatch summary for short read*

Description

Showing only mismatch read counts as color coded stacked barchart or with coverage for background.

Usage

```
plotMismatchSum(obj, show.coverage = TRUE)
```

Arguments

`obj`	obj should be a `GRanges` object, usually returned by `pileupGRAngesAsVariantTable` function. Or is a `GRanges` object with arbitrary column: read, ref, count, depth, match.
`show.coverage`	A logical value indicate whether to show the coverage or not. Default is `TRUE`.

Details

Color for DNA bases are specified in the biovizBase package and retrieved in the options. Please see `getBioColor` for more information.

Value

A `ggplot` object.

Author(s)

Tengfei Yin

Examples

```
## Not run:
data(genesymbol)
library(BSgenome.Hsapiens.UCSC.hg19)
bamfile <- system.file("extdata", "SRR027894subRBM17.bam", package="biovizBase")
gr <- GRanges("chr10", IRanges(6134000, 6135000))
test <- pileupAsGRanges(bamfile, region = gr)
test.match <- pileupGRangesAsVariantTable(test, Hsapiens)
## use plotMismatchSum directly
p <- plotMismatchSum(test.match, FALSE)
p <- plotMismatchSum(test.match, TRUE)
## use qplot generic function
p <- qplot(test.match, geom = "mismatch.summary")
p <- qplot(test.match, geom = "mismatch.summary", show.coverage = FALSE)

library(Rsamtools)

## for character
p <- qplot(bamfile, which = gr, bsgenome = Hsapiens,
      geom = "mismatch.summary", show.coverage = TRUE)
p <- qplot(bamfile, which = gr, bsgenome = Hsapiens,
      geom = "mismatch.summary", show.coverage = FALSE)

## for BamFile
p <- qplot(BamFile(bamfile), which = gr, bsgenome = Hsapiens,
      geom = "mismatch.summary", show.coverage = TRUE)
p <- qplot(BamFile(bamfile), which = gr, bsgenome = Hsapiens,
      geom = "mismatch.summary", show.coverage = FALSE)

## End(Not run)
```

---

plotOverview *Plot stacked overview for genome*

Description

Plot stacked overview for genome with or without cytoband.

Usage

```
plotOverview(obj, cytoband=FALSE)
```

Arguments

obj	A GenomicRanges object, which include extra information about cytoband.
cytoband	Logical value. Default is FALSE. If TRUE, plotting cytoband.

Details

This function requires two column of the `gieStain` and `name`. Which you could get from `getIdeogram` function in pacakge `biovizBase`.

Value

A `ggplot` object.

Author(s)

Tengfei Yin

Examples

```
library(GenomicRanges)
data(hg19IdeogramCyto)
## make shorter and clean labels
old.chrs <- seqnames(seqinfo(hg19IdeogramCyto))
new.chrs <- gsub("chr", "", old.chrs)
names(new.chrs) <- old.chrs
new.ideo <- renameSeqlevels(hg19IdeogramCyto, new.chrs)
## with cytoband
p <- plotOverview(new.ideo, cytoband = TRUE)
print(p)
## with nocytoband
p <- plotOverview(new.ideo, cytoband = FALSE)
print(p)
```

---

plotRangesLinkedToData

Plot Ranges Linked with Data

Description

Plot GRanges object structure and linked to a even spaced paralell coordinates plot which represting the data in elementeMetadata.

Usage

```
plotRangesLinkedToData(data, stat.col, stat.label, ..., annotation = list(),
                       width.ratio = 0.8,
                       heights = c(400, 100, 100, rep(300, length(annotation))))
```

Arguments

`data`	GRanges object with a DataFrame as elementMetadata.
`stat.col`	integer (variable position starting in DataFrame of data, start from 1) or strings (variable names) which indicate the column names.
`stat.label`	Labels of the columns, if missing, use stat.col.
`...`	Parameters passed to qplot.
`annotation`	A list of ggplot object.
`width.ratio`	Control the segment length of statistic layer.
`heights`	Heights of each track.

Details

Inspired by some graphics produced in some other packages, for example in package DEXseq, the author provides graphics with gene models and linked to an even spaced statistics summary. This is useful because we always plot everything along the genomic coordinates, but genomic features like exons are not evenly distributed, so we could actually treat the statistics associated with exons like categorical data, and show them as "Paralell Coordinates Plots". This is one special layout which represent the data in a nice manner and also keep the genomic structure information. With abliity of `tracks`, it's possible to generate such type of a graphic along with other annotations.

The data we want is a normal `GRanges` object, and make sure the intervals are not overlaped with each other(currently), and you may have multiple columns which store the statistics for multiple samples, then we produce the graphic we introduced above and users could pass other annotation track in the function which will be shown below the main linked track.

The reason you need to pass annotation into the function instead of binding them by `tracks` later is because binding manually with annotation tracks is tricky and this function doesn't return a ggplot object.

Value

return a frame grob; side-effect (plotting) if plot=T.

Author(s)

Tengfei Yin

Examples

```
## Not run:
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol)
txdb <- Hsapiens_UCSC_hg19_knownGene_TxDb
model <- exonsBy(txdb, by = "tx")
model17 <- subsetByOverlaps(model, genesymbol["RBM17"])
exons <- exons(txdb)
exon17 <- subsetByOverlaps(exons, genesymbol["RBM17"])
## reduce to make sure there is no overlap
## just for example
exon.new <- reduce(exon17)
## suppose
values(exon.new)$sample1 <- rnorm(length(exon.new), 10, 3)
values(exon.new)$sample2 <- rnorm(length(exon.new), 10, 10)
values(exon.new)$score <- rnorm(length(exon.new))
plotRangesLinkedToData(exon.new, stat.col = c("sample1", "sample2"))
plotRangesLinkedToData(exon.new, stat.col = 1:2)
plotRangesLinkedToData(exon.new, stat.col = 1:2, annotation = list(p))

## End(Not run)
```

`plotSingleChrom` *Plot single chromosome with cytoband*

Description

Plot single chromosome with cytoband

Usage

```
plotSingleChrom(obj, subchr, zoom.region, xlabel = FALSE)
```

Arguments

obj	A GenomicRanges object, which include extra information about cytoband.
subchr	A single character of chromosome names to show.
zoom.region	A numeric vector of length 2 indicating zoomed region.
xlabel	A logical value. Show the x label or not.

Details

User could provide the whole ideogram and use subchr to point to particular chromosome.

Value

A ggplot object.

Author(s)

Tengfei Yin

Examples

```
data(hg19IdeogramCyto)
library(grid)
vp1 <- viewport(width = 1, height = 0.14)
p <- plotSingleChrom(hg19IdeogramCyto, subchr = "chr1")
print(p, vp = vp1)
```

---

plotSpliceSum *Plot Splice Summary from RNA-seq data*

Description

Plot splice summary by simply counting overlaped junction read in weighted way or not.

Usage

```
## For character,GRangesList
## S4 method for signature 'character,GRangesList'
plotSpliceSum(data, model, ..., weighted = TRUE)
## For character,TranscriptDb
## S4 method for signature 'character,TranscriptDb'
plotSpliceSum(data, model, which,
      ..., weighted = TRUE)
```

Arguments

data	A character specifying the bam file path of RNA-seq data.
model	A GRangesList which represting different isoforms, or a TranscriptDb object. In the second case, users need to pass "which" argument which is a GRanges object to specify the region. And we get connonical model internally.
which	A GRanges object specifying the region you want to get model from the TranscriptDb object.
weighted	If TRUE, weighted by simply add 1/cases matched to each model and if FALSE, simply add 1 to every case.
...	Extra arugments passed to qplot function. such as, offset which control the height of chevron.

Details

Internally we use biovizBase:::spliceSummary for simple counting, but we encourage users to use their own robust way to make slicing summary and store it as GRangesList, then plot the summary by qplot function.

Value

A ggplot object.

Author(s)

Tengfei Yin

See Also

qplot

Examples

```
## Not run:
bamfile <- system.file("extdata", "SRR027894subRBM17.bam", package="biovizBase")
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- Hsapiens_UCSC_hg19_knownGene_TxDb
data(genesymbol)
exons <- exonsBy(txdb, by = "tx")
exons.rbm17 <- subsetByOverlaps(exons, genesymbol["RBM17"])
plotSpliceSum(bamfile, exons.rbm17)
plotSpliceSum(bamfile, exons.rbm17, weighted = FALSE, offset = 0.01)
plotSpliceSum(bamfile, txdb, which = genesymbol["RBM17"])
plotSpliceSum(bamfile, txdb, which = genesymbol["RBM17"], offset = 0.01)
plotSpliceSum(bamfile, txdb, which = genesymbol["RBM17"],
              show.label = TRUE,
              label.type = "count")

## End(Not run)
```

qplot *Generic qplot function*

Description

To visualize different objects describing biological data, we develop this generic function, and developed new types of geoms to each one. Try to make simple API and following the grammar of graphics, use higher level graphic package like ggplot2 to produce high quality graphics.

Usage

```
## For object data.frame
## S4 method for signature 'data.frame'
qplot(data, ...)

## For object matrix
## S4 method for signature 'matrix'
qplot(data, ...)

## For object numeric
## S4 method for signature 'numeric'
qplot(data, ...)

## For object numeric
## S4 method for signature 'integer'
qplot(data, ...)

## For object GRanges
## S4 method for signature 'GRanges'
qplot(data, x, y,...,
      facet_gr, legend = TRUE,
      show.coverage = TRUE,
      show.gaps = FALSE,
      show.label = FALSE,
      geom = c("full", "line","point",
               "segment",
               "coverage.line", "coverage.polygon"))

## For object GRangesList
## S4 method for signature 'GRangesList'
qplot(data, ..., freq, show.label = FALSE,
      show.gaps = TRUE, scale.size = c(5, 17),
      label.type = c("name", "count"),
      label.size = 5, label.color = "black")

## For object IRanges
## S4 method for signature 'IRanges'
qplot(data, ..., legend = TRUE,
      geom = c("full", "segment",
               "coverage.line", "coverage.polygon"))
```

```
## For object GappedAlignments
## S4 method for signature 'GappedAlignments'
qplot(data, ..., which,
      geom = c("gapped.pair", "full"),
      show.junction = FALSE)

## For object BamFile
## S4 method for signature 'BamFile'
qplot(data, ..., which,
      bsgenome, resize.extra = 10, show.coverage = TRUE,
      geom = c("gapped.pair", "full",
               "coverage.line",
               "coverage.polygon", "mismatch.summary"))

## For object character
## S4 method for signature 'character'
qplot(data, ..., which,
      bsgenome, resize.extra = 10, show.coverage = TRUE,
      geom = c("gapped.pair", "full",
               "coverage.line", "coverage.polygon",
               "mismatch.summary"))

## For object TranscriptDb
## S4 method for signature 'TranscriptDb'
qplot(data, which, ...,
      geom = c("full", "single", "tx"))

## For object BSgenome
## S4 method for signature 'BSgenome'
qplot(data, name, ...,
      geom = c("text", "segment", "point", "rectangle"))

## For object Rle
## S4 method for signature 'Rle'
qplot(data, lower, ...,
      size, shape, color, alpha,
      xlab = "x", ylab = "y",
      geom = c("point", "line", "segment"),
      type = c("raw", "viewMaxs","viewMins",
               "viewSums", "viewMeans"))

## For object RleList
## S4 method for signature 'RleList'
qplot(data, lower, ...,
      size, shape, color, alpha,
      facetByRow = TRUE,
      xlab = "x", ylab = "y",
      geom = c("point", "line", "segment"),
      type = c("raw", "viewMaxs","viewMins",
               "viewSums", "viewMeans"))
```

Arguments

data	A data.frame, matrix, GRanges, BSgenome, TranscriptDb, GappedAlignments object or any other objects for which the qplot method is defined.
x	x value, start/end/midpoint without quotes. e.g x = start, default use the midpoint.
y	y value, only be used in geom: point/line. Should be a single name of the column names in the elementMetatata without quotes. e.g y = score
geom	Geom to use (Single character for now). Please see section Geometry for details.
facet_gr	A GRanges object which contains the regions you want to facet on.
size	Size for point or lines. Could equal a column name or a fixed number. When it's fixed, please use I() to wrap the value.
shape	Shape for point or lines. Could equal a column name or a fixed number. When it's fixed, please use I() to wrap the value.
color	Color for point for lines. Could equal a column name or a fixed character. When it's fixed, please use I() to wrap the value.
alpha	Alpha blending. Could equal a column name or a fixed number. When it's fixed, please use I() to wrap the value.
lower	When object is Rle/RleList, and use other types of methods which is not "raw", at least a lower parameters which passed to slice function is required.
name	Passed to getSeq in BSgenome package. A character vector containing the names of the sequences in 'x' where to get the subsequences from, or a GRanges object, or a RangedData object, or a named RangesList object, or a named Ranges object. The RangesList or Ranges object must be named according to the sequences in 'x' where to get the subsequences from. If 'names' is missing, then 'seqnames(x)' is used.
legend	A logical value indicates whether to show legend or not. Default is TRUE
which	A GRanges object to subset the result, usually passed to the ScanBamParam function.
show.junction	A logical value indicates whether to show the line between junction reads or not.
show.coverage	A logical value indicates whether to show coverage or not. This is used for geom "mismatch.summary".
show.gaps	A logical value indicates whether to show gaps or not. This is used for geom "full".
show.label	A logical value indicates whether to show labels or not. This is used for geom "full".
freq	A numeric vector indicating counts. This used for GRangesList plot when you try to produce a summary for gene model or alternative splicing, the name of the vector corresponding to the names of the GRangesList.
resize.extra	A numeric value used to add buffer to intervals to compute stepping levels on.
bsgenome	A BSgenome object. Only need for geom "mismatch.summary".
xlab	x label.
ylab	y label.

facetByRow	A logical value, default is TRUE ,facet RleList by row. If FALSE, facet by column.
type	For Rle/RleList, "raw" plot everything, so be careful, that would be pretty slow if you have too much data. For "viewMins", "viewMaxs", "viewMeans", "viewSums", require extra arguments to slice the object. so users need to at least provide lower, more details and control please refer the the manual of slice function in IRanges. For "viewMins", "viewMaxs", we use viewWhichMin and viewWhichMax to get x scale, for "viewMeans", "viewSums", we use window midpoint as x.
scale.size	A numeric vector of length two specifying the size of point from minimum to maximum.
label.type	Either "name" or "count". "name" try to use the name of list to label the model, but "count" use information from freq to label it.
label.size	Size of label.
label.color	Color of label.
...	Extra parameters. Usually are those parameters used in qplot to control aesthetics or geometries.

Value

A ggplot object, so you can use common features from ggplot2 package to manipulate the plot.

Introduction

qplot is redefined as generic s4 method inside this package, user could use qplot in the way they are familiar with, and we are also setting limitation inside this package, like

- scales X scales is always genomic coordinates in most cases, x could be specified as start/end/midpoint when it's special geoms for interval data like point/line
- colors Try to use default color scheme defined in biovizBase package as possible as it can

Geometry

We have developed new geom for different objects, some of them may require extra parameters you need to provide. Some of the geom are more like geom + stat in ggplot2 package. e.g. "coverage.line" and "coverage.polygon".We simply combine them together, but in the future, we plan to make it more general.

This package is designed for only biological data, especially genomic data if users want to explore the data in a more flexible way, you could simply coerce the GRanges to a data.frame, then just use formal qplot function in ggplot2, or qplot generic for data.frame.

Some objects share the same geom so we introduce all the geom together in this section

Showing all the intervals as stepped rectangle, colored by strand automatically.

For TranscripDb object, showing full model.

segment Showing all the intervals as stepped segments, colored by strand automatically.

For object BSgenome, show nucleotides as colored segment.

For Rle/RleList, show histogram-like segments.

line Showing interval as line, the interval data could also be just single position when start = end, x is one of start/end/midpoint, y value is unquoted name in elementMetadata column names. y value is required.

point Showing interval as point, the interval data could also be just single position when start = end, x is one of start/end/midpoint, y value is unquoted name in elementMetadata column names. y value is required.

For object `BSgenome`, show nucleotides as colored point.

coverage.line Coverage showing as lines for interval data.

coverage.polygon Coverage showing as polygon for interval data.

splice Splicing summary. The size and width of the line and rectangle should represent the counts in each model. Need to provide model.

single For `TranscripDb` object, showing single(reduced) model only.

tx For `TranscripDb` object, showing transcirpts isoforms.

gapped.pair Show `GappedAlignments` as special stepping plots, it make sure all the reads of the same pair or qname shown in the same level and nothing falls in between. Then you can use `show.junction` arguments show the junction as lines between junction reads if any.

mismatch.summary Showing color coded mismatched stacked bar to indicate the proportion of mismatching at each position, the reference is set to gray.

text For object `BSgenome`, show nucleotides as colored text.

rectangle For object `BSgenome`, show nucleotides as colored rectangle.

Faceting

Faceting in ggbio package is a little differnt from ggplot2 in several ways

- The faceted column could only be seqnames or regions on the genome. So we limited the formula passing to facet argument, e.g something \~ seqnames, is accepted formula, you can change "something" to variable name in the elementMetadata. But you can not change the second part.
- Sometime, we need to view different regions, so we also have a facet_gr argument which accept a `GRanges`. If this is provided, it will override the default seqnames and use provided region to facet the graphics, this might be useful for different gene centric views.

Author(s)

Tengfei Yin

Examples

```
## Not run:
library(ggbio)

## overide qplot
qplot(data = mtcars, mpg, cyl)
qplot(1:3)
qplot(volcano)
qplot(c(1, 2.2, 3.3))
ggplot2::qplot(1:3)
ggplot2::qplot(c(1, 2.2, 3.3))
ggplot2::qplot(volcano)


##  GRanges
set.seed(1)
N <- 1000
```

```
library(GenomicRanges)
gr <- GRanges(seqnames =
              sample(c("chr1", "chr2", "chr3"),
                     size = N, replace = TRUE),
              IRanges(
                      start = sample(1:300, size = N, replace = TRUE),
                      width = sample(70:75, size = N,replace = TRUE)),
              strand = sample(c("+", "-", "*"), size = N,
                replace = TRUE),
              value = rnorm(N, 10, 3), score = rnorm(N, 100, 30),
              group = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

qplot(gr)
qplot(gr, geom = "full")
qplot(gr, geom = "segment")
qplot(gr, geom = "line", y = value)
qplot(gr, geom = "point", y = value)
qplot(gr, geom = "coverage.line")
qplot(gr, geom = "coverage.polygon")

gr.sub <- gr[seqnames(gr) == "chr1"] #or
p1 <- qplot(gr.sub, geom = "full") + opts(title = "full")
p2 <- qplot(gr.sub, geom = "point", y = value) + opts(title = "point")
p3 <- qplot(gr.sub, geom = "line", y = value) + opts(title = "line")
p4 <- qplot(gr.sub, geom = "coverage.line") + opts(title = "coverage.line")
p5 <- qplot(gr.sub, geom = "coverage.polygon") + opts(title = "coverage.polygon")
library(gridExtra)
grid.arrange(p1, p2, p3, p4, p5, ncol = 2)

qplot(gr, ncol = 2)
## faceting, use facets not facet
qplot(gr, facets = group ~ seqnames)
qplot(gr, geom = "segment", facets = group ~ seqnames)
qplot(gr, geom = "line", y = value, facets = group ~ seqnames)
qplot(gr, geom = "point", y = value, facets = group ~ seqnames)
qplot(gr, geom = "coverage.line", facets = group ~ seqnames)
qplot(gr, geom = "coverage.polygon", facets = group ~ seqnames)

## facet gr
gr.region <- GRanges(c("chr1", "chr2", "chr3"),
                     IRanges(c(100, 200, 250),
                             width = 70))
## facet_grid
qplot(gr, facet_gr = gr.region)
## facet_wrap
qplot(gr, facet_gr = gr.region, nrow = 2) +
  scale_y_continuous(limits = c(0, 90))


## checvron
gr <- GRanges("chr1", IRanges(c(100, 200, 300), width = 50))
p <- qplot(gr)
gr.gaps <- gaps(gr)[-1]
values(gr.gaps)$score <- c(1, 100)
```

```
p1 <- p + geom_chevron(gr.gaps)
p2 <- p + geom_chevron(gr.gaps, aes(size = score), offset = "score",
                 chevron.height = c(0.1, 0.2))
p3 <- p + geom_chevron(gr.gaps, offset = -0.1)
tracks(p1, p2, p3)

## GRangesList
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
exons.tx <- exonsBy(txdb, by = "tx")
exons.rbm17 <- subsetByOverlaps(exons.tx, genesymbol["RBM17"])
nms <- names(exons.rbm17)
freqs <- c(100, 200, 300)
names(freqs) <- nms
p.splice1 <- qplot(exons.rbm17)
## when turning on frequency
p.splice <- qplot(exons.rbm17, freq = freqs, show.label = TRUE, label.type = "count",
      scale.size = c(1, 5), label.size = 3)
p.splice2 <- qplot(exons.rbm17, freq = freqs, show.label = TRUE, offset = 0.05,
                   label.type = "count")
print(p.splice1)
print(p.splice2)

ir <- IRanges(c(10, 20, 30) ,width  = 15)
qplot(ir)
ir <- ranges(gr[seqnames(gr) == "chr1"])[1:40]
p1 <- qplot(ir) + opts(title = "full")
p2 <- qplot(ir, geom = "segment")+ opts(title = "segment")
p3 <- qplot(ir, geom = "coverage.line")+ opts(title = "coverage.line")
p4 <- qplot(ir, geom = "coverage.polygon")+ opts(title = "coverage.polygon")
library(gridExtra)
grid.arrange(p1, p2, p3, p4, ncol = 2)

library(IRanges)
set.seed(1)
lambda <- c(rep(0.001, 4500), seq(0.001, 10, length = 500),
            seq(10, 0.001, length = 500))
xVector <- rpois(1e4, lambda)
xRle <- Rle(xVector)
xRleList <- RleList(xRle, 2L * xRle)


qplot(xRle)
qplot(xRle, geom = "line")
qplot(xRle, geom = "segment")
qplot(xRle, type = "viewMaxs", lower = 5)
qplot(xRle, type = "viewMins", lower = 5)
qplot(xRle, type = "viewMeans", lower = 5)
qplot(xRle, type = "viewSums", lower = 5)

qplot(xRleList)
qplot(xRleList, geom = "segment")
qplot(xRleList, geom = "line")
qplot(xRleList, type = "viewMaxs", lower = 5)
qplot(xRleList, type = "viewMaxs", lower = 5, geom = "line")
qplot(xRleList, type = "viewSums", lower = 5, geom = "segment",
```

```
      facetByRow = FALSE, color = I("red"), size = I(5))
qplot(xRle, size = y)
qplot(xRle, type = "viewSums", lower = 5)

qplot(xRle, type = "viewSums", lower = 5, size = I(10), color = I("red"),
      alpha = y)

## End(Not run)
```

`rescale` *rescale ggplot object*

Description

Rescale a numeric vector or ggplot object, could be used for static zoom-in in ggbio.

Usage

```
## For signature numeric
## S4 method for signature 'numeric'
rescale(x, to = c(0, 1),
       from = range(x, na.rm = TRUE))

## For signature ggplot
## S4 method for signature 'ggplot'
rescale(x, xlim, ylim, sx = 1, sy = 1)
```

Arguments

`x`	A numeric object or ggplot object to be rescaled.
`to`	For numeric object. it's a vector of two numeric values, specifying the range to be rescale.
`from`	Range of x.
`xlim`	For ggplot object. This specify the new limits on x-scale.
`ylim`	For ggplot object. This specify the new limits on y-scale.
`sx`	Scale fold for x-scale. Default is 1, no change.
`sy`	Scale fold for y-scale. Default is 1, no change.

Details

When `x` is numeric value, it's just call scales::rescale, please refer to the manual page to check more details. If `x` is ggplot object, it first try to estimate current x limits and y limits of the ggplot object, then rescale based on those information.

Value

Return the object of the same class as `x` after rescaling.

Author(s)

Tengfei Yin

Examples

```
## Not run:
library(ggbio)
head(mtcars)
range(mtcars$mpg)
p <- qplot(data = mtcars, x = mpg, y = disp, geom = "point")
p.new <- rescale(p, xlim = c(20, 25))

## End(Not run)
```

---

tracks *Tracks for genomic graphics*

Description

In most genome browsers, they all have such a view that including many tracks, could be any anntation data along genomic coordinate. So we try to provide a convenient constructor for building tracks, which here in this package is simply vertically binding of several plots. It's essentially a `grid.arrange`. So if users want to have more delicate control over their tracks, they need manipulate the graphics in ggplot2 level or grid levels.

Usage

```
tracks(..., show.axis.text.y = FALSE,
               show.ticks = FALSE,
               show.title = TRUE,
               legend = FALSE, xlim, ylim)
```

Arguments

`...`	Plots of class ggplot2, trellis, or grobs, and valid arguments to grid.layout.
`show.axis.text.y`	logical value indicate to show y axis text or not.
`show.ticks`	logical value indicate to show ticks or not.
`show.title`	logical value indicate to show title of tracks or not.
`legend`	Default is FALSE, remove legend, this make sure all tracks are aligned exactly based on the same position.
`xlim`	Limits on x.
`ylim`	Limits on y.

Details

`tracks` function has some extra features and limitations compare to `grid.arrange`.

- Always sitting on genomic or protein space.
- Always using ncol = 1 as default arguments.
- For now, since the unbalanced legend and labels in ggplot2 has been solved (maybe just I haven't found such features). We simply remove legend and y axis labels to make sure all tracks are aligned exactly in the same way.

- Remove the x-axis for most track except the last one.
- Does the ajustment of margins for you automatically.
- Doesn't like `qplot`, tracks doesn't return `ggplot` object. so processing your plot before you pass them to `tracks`.
- Tracks cannot guarantee all trakcs are aligned well in all the cases sinmly because control of grobs sometimes are tricky and based on what users passed. Like theme change, or uneven labels or legend could affact the alignemnts. Eventhough I tried hard to align them in most cases.
- When y axis text length are not equal across tracks, it's hard to align. So you will see small wiggles. Even missing y label would affect the accurate alignmetns too.
- So by default, we remove y axis text and legend to make sure they are aligned exactly at the same position. Clearly this is not a good practice.
- We will try our best to fix this restriction in the future release. So advanced users could use gridExtra package to modify the graphics before passing to `grid.arrange`.

Value

return a frame grob; side-effect (plotting) if plot=T.

Author(s)

Tengfei Yin

See Also

`grid.arrange`

Examples

```
## Not run:
library(BSgenome.Hsapiens.UCSC.hg19)
gr <- GRanges("chr1", IRanges(5e7, 5e7+50))
p1 <- qplot(Hsapiens, name = gr, geom = "text")
p2 <- qplot(Hsapiens, name = gr, geom = "point")
p3 <- qplot(Hsapiens, name = gr, geom = "segment")
p4 <- qplot(Hsapiens, name = gr, geom = "rectangle")
tracks(p1, p2, p3, p4)

## End(Not run)
```

Index