Package 'ggbio'

October 9, 2013

Version 1.8.8

Title Visualization tools for genomic data.

Description The ggbio package extends and specializes the grammar of graphics for biological data. The graphics are designed to answer common scientific questions, in particular those often asked of high throughput genomics data. All core Bioconductor data structures are supported, where appropriate. The package supports detailed views of particular genomic regions, as well as genome-wide overviews. Supported overviews include ideograms and grand linear views. High-level plots include sequence fragment length, edge-linked interval to data view, mismatch pileup, and several splicing summaries.

Author Tengfei Yin, Dianne Cook, Michael Lawrence

Maintainer Tengfei Yin <yintengfei@gmail.com>

Depends methods, ggplot2 (>= 0.9.2)

Imports methods, biovizBase(>= 1.7.8), reshape2, gtable, ggplot2(>= 0.9.2), BiocGenerics, Biobase, IRanges, GenomicRanges,GenomicFeatures, Rsamtools, BSgenome, gridExtra, scales, plyr,VariantAnnotation, Hmisc, rtracklayer

Suggests BSgenome.Hsapiens.UCSC.hg19,TxDb.Hsapiens.UCSC.hg19.knownGene, affyPLM, chipseq,TxDb.Mmusculus.UCSC.mm9.knownGene, knitr

biocViews Infrastructure, Visualization, Bioinformatics

URL http://tengfei.github.com/ggbio/

BugReports https://github.com/tengfei/ggbio/issues

License Artistic-2.0

LazyLoad Yes

Collate utils.R AllGenerics.R ggbio-class.R ggplot-method.R theme.R tracks.R mold-method.R geom_chevron-method.R geom_alignment-method.R geom_arch-method.R geom_arrow-method.R geom_arrowrect-method.R geom_rect-method.R geom_segment-method.R geom_bar-method.R layout_circle-method.R

layout_karyogram-method.R layout_linear-method.R
stat_aggregate-method.R stat_coverage-method.R
stat_identity-method.R stat_mismatch-method.R
stat_stepping-method.R stat_gene-method.R stat_table-method.R
stat_bin-method.R stat_slice-method.R stat_reduce-method.R
coord_genome-method.R autoplot-method.R hack.R ideogram.R
plotGrandLinear.R plotRangesLinkedToData.R
plotFragLength-method.R plotSpliceSum-method.R rescale-method.R zzz.R

R topics documented:

`arrangeGrobByParsingLegend` *Arrange grobs by parse their legend.*

Description

Arrange grobs and parse their legend, then put it together on the right.

Usage

```
arrangeGrobByParsingLegend(..., nrow = NULL, ncol = NULL,
                           widths = c(4, 1), legend.idx = NULL)
```

Arguments

`...`	ggplot graphics.
`nrow`	number of row for layout.
`ncol`	number of columns for layout
`widths`	width ratio for plot group and legend group.
`legend.idx`	legend index you want to keep.

Value

a

Author(s)

Tengfei Yin

Examples

```
library(ggplot2)
p1 <- qplot(x = mpg, y= cyl, data = mtcars, color = carb)
p2 <- qplot(x = mpg, y= cyl, data = mtcars, color = wt)
p3 <- qplot(x = mpg, y= cyl, data = mtcars, color = qsec)
p4 <- qplot(x = mpg, y= cyl, data = mtcars, color = gear)
arrangeGrobByParsingLegend(p1, p2, p3, p4)
arrangeGrobByParsingLegend(p1, p2, p3, p4, ncol = 1)
arrangeGrobByParsingLegend(p1, p2, p3, p4, legend.idx = 2)
```

autoplot *Generic autoplot function*

Description

autoplot is a generic function to visualize various data object, it tries to give better default graphics and customized choices for each data type, quick and convenient to explore your genomic data compare to low level ggplot method, it is much simpler and easy to produce fairly complicate graphics, though you may lose some flexibility for each layer.

Usage

```
## S4 method for signature 'GRanges'
autoplot(object, ..., chr, xlab, ylab, main, truncate.gaps = FALSE,
                 truncate.fun = NULL, ratio = 0.0025, space.skip = 0.1,
                 legend = TRUE, geom = NULL, stat = NULL,
                 chr.weight = NULL,
                 coord = c("default", "genome", "truncate_gaps"),
                 layout = c("linear", "karyogram", "circle"))


## S4 method for signature 'GRangesList'
autoplot(object, ..., xlab, ylab, main, indName = "grl_name",
                 geom = NULL, stat = NULL, type = c("none", "sashimi"),
                 coverage.col = "gray50", coverage.fill = coverage.col,
                 group.selfish = FALSE, arch.offset = 1.3)


## S4 method for signature 'IRanges'
autoplot(object, ..., xlab, ylab, main)

## S4 method for signature 'Seqinfo'
autoplot(object, ideogram = FALSE, ... )

## S4 method for signature 'GappedAlignments'
autoplot(object, ..., xlab, ylab, main, which,
      geom = NULL, stat = NULL)

## S4 method for signature 'BamFile'
autoplot(object, ..., which, xlab, ylab, main,
              bsgenome, geom = "line", stat = "coverage",
              method = c("estimate", "raw"),
              coord = c("linear", "genome"), resize.extra = 10,
              space.skip = 0.1, show.coverage = TRUE)

## S4 method for signature 'character'
autoplot(object, ..., xlab, ylab, main, which, asRangedData = FALSE)
```

```
## S4 method for signature 'TranscriptDb'
autoplot(object, which, ..., xlab, ylab, main, truncate.gaps =
                 FALSE, truncate.fun = NULL, ratio = 0.0025, geom =
                 c("alignment"), stat = c("identity", "reduce"),
                 names.expr = "tx_name(gene_id)")



## S4 method for signature 'BSgenome'
autoplot(object, which, ...,
                    xlab, ylab, main, geom = c("text",
                   "segment", "point", "rect"))

## S4 method for signature 'Rle'
autoplot(object, ..., xlab, ylab, main, binwidth, nbin = 30,
          geom = NULL, stat = c("bin", "identity", "slice"),
          type = c("viewSums", "viewMins", "viewMaxs", "viewMeans"))

## S4 method for signature 'RleList'
autoplot(object, ..., xlab, ylab, main, nbin = 30, binwidth,
         facetByRow = TRUE, stat = c("bin", "identity", "slice"),
         geom = NULL, type = c("viewSums", "viewMins", "viewMaxs", "viewMeans"))

## S4 method for signature 'matrix'
autoplot(object, ..., xlab, ylab, main,
              geom = c("tile", "raster"), axis.text.angle = NULL,
              hjust = 0.5, na.value = NULL,
              rownames.label = TRUE, colnames.label = TRUE,
              axis.text.x = TRUE, axis.text.y = TRUE)


## S4 method for signature 'ExpressionSet'
autoplot(object, ..., type = c("heatmap", "none",
                 "scatterplot.matrix", "pcp", "MA", "boxplot",
                 "mean-sd", "volcano", "NUSE", "RLE"), test.method =
                 "t", rotate = FALSE, pheno.plot = FALSE, main_to_pheno
                 = 4.5, padding = 0.2)


## S4 method for signature 'SummarizedExperiment'
autoplot(object, ..., type = c("heatmap", "link", "pcp",
                 "boxplot", "scatterplot.matrix"), pheno.plot = FALSE,
                 main_to_pheno = 4.5, padding = 0.2, assay.id = 1)


## S4 method for signature 'VCF'
autoplot(object, ...,
```

```
xlab, ylab, main,
assay.id,
type = c("geno", "info", "fixed"),
full.string = FALSE,
ref.show = TRUE,
genome.axis = TRUE,
transpose = TRUE)
```

Arguments

object	object to be plot.
ideogram	Weather to call plotIdeogram or not, default is FALSE, if TRUE, layout_karyogram will be called.
transpose	logical value, defaut TRUE, always make features from VCF as x, so we can use it to map to genomic position.
axis.text.angle	axis text angle.
axis.text.x	logical value indicates whether to show x axis and labels or not.
axis.text.y	logical value indicates whether to show y axis and labels or not.
hjust	horizontal just for axis text.
rownames.label	logical value indicates whether to show rownames of matrix as y label or not.
colnames.label	logical value indicates whether to show colnames of matrix as y label or not.
na.value	color for NA value.
rotate	
pheno.plot	show pheno plot or not.
main_to_pheno	main matrix plot width to pheno plot width ratio.
padding	padding between plots.
assay.id	index for assay you are going to use.
geom	Geom to use (Single character for now). Please see section Geometry for details.
truncate.gaps	logical value indicate to truncate gaps or not.
truncate.fun	shrinkage function. Please see shrinkagefun in package biovizBase.
ratio	used in maxGap.
space.skip	space ratio between chromosome spaces in coordate genome.
coord	Coodinate system.
chr.weight	numeric vectors which sum to <1, the names of vectors has to be matched with seqnames in seqinfo, and you can only specify part of the seqnames, other lengths of chromosomes will be assined proportionally to their seqlengths, for example, you could specify chr1 to be 0.5, so the chr1 will take half of the space and other chromosomes squeezed to take left of the space.
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.coverage	A logical value indicates whether to show coverage or not. This is used for geom "mismatch.summary".
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. For ExpreesionSet, ploting types.
layout	Layout including linear, circular and karyogram. for GenomicRangesList, it only supports circular layout.
method	method used for parsing coverage from bam files. 'estimate' use fast esitmated method and 'raw' use relatively slow parsing method.
asRangedData	when object is character, it may be imported by import function in package rtracklayer, then asRangedData passed to import function. If FALSE, coerce object to GRanges.
test.method	test method
...	Extra parameters. Usually are those parameters used in autoplot to control aesthetics or geometries.
main	title.
stat	statistical transformation.
indName	When coerce GRangesList to GRanges, names created for each group.
coverage.col	coverage stroke color.
coverage.fill	coverage fill color.
group.selfish	Passed to addStepping, control whether to show each group as unique level or not. If set to FALSE, if two groups are not overlapped with each other, they will probably be layout in the same level to save space.
arch.offset	arch.offset.
names.expr	names expression used for creating labels.
binwidth	width of the bins.
nbin	number of bins.
genome.axis	logical value, if TRUE, whenever possible, try to parse genomic postition for each column(e.g. SummarizedExperiment), show column as exatcly the genomic position instead of showing them side by side and indexed from 1.
full.string	logical value. If TRUE, show full string of indels in plot for VCF.
ref.show	logical value. If TRUE, show REF in VCF at bottom track.
chr	characters indicates the seqnames to be subseted.

Value

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

Introduction

autoplot is redefined as generic s4 method inside this package, user could use autoplot 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 autoplot function in ggplot2, or autoplot 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.

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

```
library(ggbio)

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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

idx <- sample(1:length(gr), size = 50)


##################################################
### code chunk number 3: default
##################################################
autoplot(gr[idx])


##################################################
### code chunk number 4: bar-default-pre
##################################################
set.seed(123)
gr.b <- GRanges(seqnames = "chr1", IRanges(start = seq(1, 100, by = 10),
                  width = sample(4:9, size = 10, replace = TRUE)),
```

```
                score = rnorm(10, 10, 3), value = runif(10, 1, 100))
gr.b2 <- GRanges(seqnames = "chr2", IRanges(start = seq(1, 100, by = 10),
                  width = sample(4:9, size = 10, replace = TRUE)),
                score = rnorm(10, 10, 3), value = runif(10, 1, 100))
gr.b <- c(gr.b, gr.b2)
head(gr.b)


####################################################
### code chunk number 5: bar-default
####################################################
p1 <- autoplot(gr.b, geom = "bar")
## use value to fill the bar
p2 <- autoplot(gr.b, geom = "bar", aes(fill = value))
tracks(default = p1, fill = p2)


####################################################
### code chunk number 6: autoplot.Rnw:236-237
####################################################
autoplot(gr[idx], geom = "arch", aes(color = value), facets = sample ~ seqnames)


####################################################
### code chunk number 7: gr-group
####################################################
library(ggbio)
gra <- GRanges("chr1", IRanges(c(1,7,20), end = c(4,9,30)), group = c("a", "a", "b"))
## if you desn't specify group, then group based on stepping levels, and gaps are computed without
## considering extra group method
p1 <- autoplot(gra, aes(fill = group), geom = "alignment")
## when use group method, gaps only computed for grouped intervals.
## default is group.selfish = TRUE, each group keep one row.
## in this way, group labels could be shown as y axis.
p2 <- autoplot(gra, aes(fill = group, group = group), geom = "alignment")
## group.selfish = FALSE, save space
p3 <- autoplot(gra, aes(fill = group, group = group), geom = "alignment", group.selfish = FALSE)
tracks('non-group' = p1,'group.selfish = TRUE' = p2 , 'group.selfish = FALSE' = p3)


####################################################
### code chunk number 8: gr-facet-strand
####################################################
autoplot(gr, stat = "coverage", geom = "area",
         facets = strand ~ seqnames, aes(fill = strand))


####################################################
### code chunk number 9: gr-autoplot-circle
####################################################
autoplot(gr[idx], layout = 'circle')
```

```
##################################################
### code chunk number 10: gr-circle
##################################################
seqlengths(gr) <- c(400, 500, 700)
values(gr)$to.gr <- gr[sample(1:length(gr), size = length(gr))]
idx <- sample(1:length(gr), size = 50)
gr <- gr[idx]
ggplot() + layout_circle(gr, geom = "ideo", fill = "gray70", radius = 7, trackWidth = 3) +
  layout_circle(gr, geom = "bar", radius = 10, trackWidth = 4,
                aes(fill = score, y = score)) +
  layout_circle(gr, geom = "point", color = "red", radius = 14,
                trackWidth = 3, grid = TRUE, aes(y = score)) +
  layout_circle(gr, geom = "link", linked.to = "to.gr", radius = 6, trackWidth = 1)


##################################################
### code chunk number 11: seqinfo-src
##################################################
data(hg19Ideogram, package = "biovizBase")
sq <- seqinfo(hg19Ideogram)
sq


##################################################
### code chunk number 12: seqinfo
##################################################
autoplot(sq[paste0("chr", c(1:22, "X"))])


##################################################
### code chunk number 13: ir-load
##################################################
set.seed(1)
N <- 100
ir <-  IRanges(start = sample(1:300, size = N, replace = TRUE),
               width = sample(70:75, size = N,replace = TRUE))
## add meta data
df <- DataFrame(value = rnorm(N, 10, 3), score = rnorm(N, 100, 30),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))
values(ir) <- df
ir


##################################################
### code chunk number 14: ir-exp
##################################################
p1 <- autoplot(ir)
p2 <- autoplot(ir, aes(fill = pair)) + theme(legend.position = "none")
p3 <- autoplot(ir, stat = "coverage", geom = "line", facets = sample ~. )
p4 <- autoplot(ir, stat = "reduce")
```

```
tracks(p1, p2, p3, p4)


################################################
### code chunk number 15: grl-simul
################################################
set.seed(1)
N <- 100
## ======================================================================
##  simmulated GRanges
## ======================================================================
gr <- GRanges(seqnames =
              sample(c("chr1", "chr2", "chr3"),
                     size = N, replace = TRUE),
              IRanges(
                      start = sample(1:300, size = N, replace = TRUE),
                      width = sample(30:40, size = N,replace = TRUE)),
              strand = sample(c("+", "-", "*"), size = N,
                replace = TRUE),
              value = rnorm(N, 10, 3), score = rnorm(N, 100, 30),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))


grl <- split(gr, values(gr)$pair)


################################################
### code chunk number 16: grl-exp
################################################
## default gap.geom is 'chevron'
p1 <- autoplot(grl, group.selfish = TRUE)
p2 <- autoplot(grl, group.selfish = TRUE, main.geom = "arrowrect", gap.geom = "segment")
tracks(p1, p2)


################################################
### code chunk number 17: grl-name
################################################
autoplot(grl, aes(fill = ..grl_name..))
## equal to
## autoplot(grl, aes(fill = grl_name))


################################################
### code chunk number 18: rle-simul
################################################
library(IRanges)
library(ggbio)
set.seed(1)
lambda <- c(rep(0.001, 4500), seq(0.001, 10, length = 500),
```

```
            seq(10, 0.001, length = 500))

## @knitr create
xVector <- rpois(1e4, lambda)
xRle <- Rle(xVector)
xRle


##################################################
### code chunk number 19: rle-bin
##################################################
p1 <- autoplot(xRle)
p2 <- autoplot(xRle, nbin = 80)
p3 <- autoplot(xRle, geom = "heatmap", nbin = 200)
tracks('nbin = 30' = p1, "nbin = 80" = p2, "nbin = 200(heatmap)" = p3)


##################################################
### code chunk number 20: rle-id
##################################################
p1 <- autoplot(xRle, stat = "identity")
p2 <- autoplot(xRle, stat = "identity", geom = "point", color = "red")
tracks('line' = p1, "point" = p2)


##################################################
### code chunk number 21: rle-slice
##################################################
p1 <- autoplot(xRle, type = "viewMaxs", stat = "slice", lower = 5)
p2 <- autoplot(xRle, type = "viewMaxs", stat = "slice", lower = 5, geom = "heatmap")
tracks('bar' = p1, "heatmap" = p2)


##################################################
### code chunk number 22: rlel-simul
##################################################
xRleList <- RleList(xRle, 2L * xRle)
xRleList


##################################################
### code chunk number 23: rlel-bin
##################################################
p1 <- autoplot(xRleList)
p2 <- autoplot(xRleList, nbin = 80)
p3 <- autoplot(xRleList, geom = "heatmap", nbin = 200)
tracks('nbin = 30' = p1, "nbin = 80" = p2, "nbin = 200(heatmap)" = p3)


##################################################
### code chunk number 24: rlel-id
##################################################
p1 <- autoplot(xRleList, stat = "identity")
```

```
p2 <- autoplot(xRleList, stat = "identity", geom = "point", color = "red")
tracks('line' = p1, "point" = p2)


##################################################
### code chunk number 25: rlel-slice
##################################################
p1 <- autoplot(xRleList, type = "viewMaxs", stat = "slice", lower = 5)
p2 <- autoplot(xRleList, type = "viewMaxs", stat = "slice", lower = 5, geom = "heatmap")
tracks('bar' = p1, "heatmap" = p2)


##################################################
### code chunk number 26: txdb
##################################################
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene


##################################################
### code chunk number 27: txdb-visual
##################################################
p1 <- autoplot(txdb, which = genesymbol["ALDOA"], names.expr = "tx_name:::gene_id")
p2 <- autoplot(txdb, which = genesymbol["ALDOA"], stat = "reduce", color = "brown",
               fill = "brown")
tracks(full = p1, reduce = p2, heights = c(5, 1)) + ylab("")


##################################################
### code chunk number 28: ga-load
##################################################
library(Rsamtools)
data("genesymbol", package = "biovizBase")
bamfile <- system.file("extdata", "SRR027894subRBM17.bam", package="biovizBase")
## need to set use.names = TRUE
ga <- readBamGappedAlignments(bamfile,
                              param = ScanBamParam(which = genesymbol["RBM17"]),
                              use.names = TRUE)


##################################################
### code chunk number 29: ga-exp
##################################################
p1 <- autoplot(ga)
p2 <- autoplot(ga, geom = "rect")
p3 <- autoplot(ga, geom = "line", stat = "coverage")
tracks(default = p1, rect = p2, coverage = p3)


##################################################
### code chunk number 30: bf-load (eval = FALSE)
##################################################
```

```
## library(Rsamtools)
## bamfile <- "./wgEncodeCaltechRnaSeqK562R1x75dAlignsRep1V2.bam"
## bf <- BamFile(bamfile)


##################################################
### code chunk number 31: bf-est-cov (eval = FALSE)
##################################################
## autoplot(bamfile)
## autoplot(bamfile, which = c("chr1", "chr2"))
## autoplot(bf)
## autoplot(bf, which = c("chr1", "chr2"))
##
## data(genesymbol, package = "biovizBase")
## autoplot(bamfile,  method = "raw", which = genesymbol["ALDOA"])
##
## library(BSgenome.Hsapiens.UCSC.hg19)
## autoplot(bf, stat = "mismatch", which = genesymbol["ALDOA"], bsgenome = Hsapiens)


##################################################
### code chunk number 32: char-bam (eval = FALSE)
##################################################
## bamfile <- "./wgEncodeCaltechRnaSeqK562R1x75dAlignsRep1V2.bam"
## autoplot(bamfile)


##################################################
### code chunk number 33: char-gr
##################################################
library(rtracklayer)
test_path <- system.file("tests", package = "rtracklayer")
test_bed <- file.path(test_path, "test.bed")
autoplot(test_bed, aes(fill = name))


##################################################
###  matrix
##################################################
volcano <- volcano[20:70, 20:60] - 150
autoplot(volcano)
autoplot(volcano, xlab = "xlab", main = "main", ylab = "ylab")
## special scale theme for 0-centered values
autoplot(volcano, geom = "raster")+scale_fill_fold_change()

## when a matrix has colnames and rownames label them by default
colnames(volcano) <- sort(sample(1:300, size = ncol(volcano), replace = FALSE))
autoplot(volcano)+scale_fill_fold_change()

rownames(volcano) <- letters[sample(1:24, size = nrow(volcano), replace = TRUE)]
autoplot(volcano)

## even with row/col names, you could also disable it and just use numeric index
```

```
autoplot(volcano, colnames.label = FALSE)
autoplot(volcano, rownames.label = FALSE, colnames.label = FALSE)

## don't want the axis has label??
autoplot(volcano, axis.text.x = FALSE)
autoplot(volcano, axis.text.y = FALSE)


# or totally remove axis
colnames(volcano) <- lapply(letters[sample(1:24, size = ncol(volcano),
replace = TRUE)],
function(x){
   paste(rep(x, 7), collapse = "")
})

## Oops, overlapped
autoplot(volcano)
## tweak with it.
autoplot(volcano, axis.text.angle =  -45, hjust = 0)

## when character is the value
x <- sample(c(letters[1:3], NA), size = 100, replace = TRUE)
mx <- matrix(x, nrow = 5)
autoplot(mx)
## tile gives you a white margin
rownames(mx) <- LETTERS[1:5]
autoplot(mx, color = "white")
colnames(mx) <- LETTERS[1:20]
autoplot(mx, color = "white")
autoplot(mx, color = "white", size = 2)
## weird in aes(), though works
## default tile is flexible
autoplot(mx, aes(width = 0.6, height = 0.6))
autoplot(mx, aes(width = 0.6, height = 0.6), na.value = "white")
autoplot(mx,  aes(width = 0.6, height = 0.6)) + theme_clear()

##################################################
### Views
##################################################
lambda <- c(rep(0.001, 4500), seq(0.001, 10, length = 500),
            seq(10, 0.001, length = 500))
xVector <- dnorm(1:5e3, mean = 1e3, sd = 200)
xRle <- Rle(xVector)
v1 <- Views(xRle, start = sample(.4e3:.6e3, size = 50, replace = FALSE), width =1000)
autoplot(v1)
names(v1) <- letters[sample(1:24, size = length(v1), replace = TRUE)]
autoplot(v1)
autoplot(v1, geom = "tile", aes(width = 0.5, height = 0.5))
autoplot(v1, geom = "line")
autoplot(v1, geom = "line", aes(color = row)) + theme(legend.position = "none")
autoplot(v1, geom = "line", facets = NULL)
autoplot(v1, geom = "line", facets = NULL, alpha  = 0.1)
```

```
#################################################
### ExpressionSet
#################################################
library(Biobase)
data(sample.ExpressionSet)
sample.ExpressionSet
set.seed(1)
## select 50 features
idx <- sample(seq_len(dim(sample.ExpressionSet)[1]), size = 50)
eset <- sample.ExpressionSet[idx,]
eset
autoplot(as.matrix(pData(eset)))

## default heatmap
p1 <- autoplot(eset)
p2 <- p1 + scale_fill_fold_change()
p2
autoplot(eset)
autoplot(eset, geom = "tile", color = "white", size = 2)
autoplot(eset, geom = "tile", aes(width = 0.6, height = 0.6))

autoplot(eset, pheno.plot = TRUE)
idx <- order(pData(eset)[,1])
eset2 <- eset[,idx]
autoplot(eset2, pheno.plot = TRUE)

## parallel coordainte plot
autoplot(eset, type = "pcp")

## boxplot
autoplot(eset, type = "boxplot")


## scatterplot.matrix
## slow, be carefull
autoplot(eset[, 1:7], type = "scatterplot.matrix")

## mean-sd
autoplot(eset, type = "mean-sd")


## volcano
autoplot(eset, type = "volcano", fac = pData(sample.ExpressionSet)$type)


#################################################
### SummarizedExperiment
#################################################
library(GenomicRanges)
nrows <- 200; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
counts2 <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
```

```
rowData <- GRanges(rep(c("chr1", "chr2"), c(50, 150)),
                   IRanges(floor(runif(200, 1e5, 1e6)), width=100),
                   strand=sample(c("+", "-"), 200, TRUE))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                     row.names=LETTERS[1:6])
sset <- SummarizedExperiment(assays=SimpleList(counts=counts,
                                               counts2 = counts2),
                             rowData=rowData, colData=colData)
autoplot(sset) + scale_fill_fold_change()
autoplot(sset, pheno.plot = TRUE)


####################################################
### pcp
####################################################
autoplot(sset, type = "pcp")


####################################################
### boxplot
####################################################
autoplot(sset, type = "boxplot")


####################################################
### scatterplot matrix
####################################################
autoplot(sset, type = "scatterplot.matrix")


####################################################
### vcf
####################################################
library(VariantAnnotation)
vcffile <- system.file("extdata", "chr22.vcf.gz", package="VariantAnnotation")
vcf <- readVcf(vcffile, "hg19")
## default use type 'geno'
## default use genome position
autoplot(vcf)
## or disable it
autoplot(vcf, genome.axis = FALSE)
## not transpose
autoplot(vcf, genome.axis = FALSE, transpose = FALSE, rownames.label = FALSE)
autoplot(vcf)
## use
autoplot(vcf, assay.id = "DS")
## equivalent to
autoplot(vcf, assay.id = 2)
## Not run:
## doesn't work when assay.id cannot find
autoplot(vcf, assay.id = "NO")

## End(Not run)
```

```
## use AF or first
autoplot(vcf, type = "info")
## geom bar
autoplot(vcf, type = "info", aes(y  = THETA))
autoplot(vcf, type = "info", aes(y  = THETA, fill = VT, color = VT))
autoplot(vcf, type = "fixed")
autoplot(vcf, type = "fixed", size = 10) + xlim(c(50310860, 50310890)) + ylim(0.75, 1.25)

p1 <- autoplot(vcf, type = "fixed") + xlim(50310860, 50310890)
p2 <- autoplot(vcf, type = "fixed", full.string = TRUE) + xlim(50310860, 50310890)
tracks("full.string = FALSE" = p1, "full.string = TRUE" = p2)+
  scale_y_continuous(breaks = NULL, limits = c(0, 3))
p3 <- autoplot(vcf, type = "fixed", ref.show = FALSE) + xlim(50310860, 50310890) +
    scale_y_continuous(breaks = NULL, limits = c(0, 2))
p3


##################################################
### code chunk number 56: bs-v
##################################################
library(BSgenome.Hsapiens.UCSC.hg19)
data(genesymbol, package = "biovizBase")
p1 <- autoplot(Hsapiens, which = resize(genesymbol["ALDOA"], width = 50))
p2 <- autoplot(Hsapiens, which = resize(genesymbol["ALDOA"], width = 50), geom = "rect")
tracks(text = p1, rect = p2)


##################################################
### code chunk number 57: sessionInfo
##################################################
sessionInfo()
```

---

geom_alignment *Alignment geoms for GRanges object*

Description

Show interval data as alignment.

Usage

```
## S4 method for signature 'GRanges'
geom_alignment(data, ..., xlab, ylab, main, facets = NULL, stat =
                 c("stepping", "identity"), range.geom = c("rect",
                 "arrowrect"), gap.geom = c("chevron", "arrow",
                 "segment"), rect.height = NULL, group.selfish = TRUE)

## S4 method for signature 'TranscriptDb'
```

```
geom_alignment(data, ..., which, xlim, truncate.gaps = FALSE,
                  truncate.fun = NULL, ratio = 0.0025, xlab, ylab, main,
                  facets = NULL, geom = "alignment",
                  stat = c("identity", "reduce"),
                  range.geom = "rect", gap.geom = "arrow",
                  utr.geom = "rect", names.expr = "tx_name(gene_id)")
```

Arguments

data	A GRanges or data.frame or TranscriptDb object.
...	Extra parameters such as aes() passed.
xlab	Label for x
ylab	Label for y
main	Title for plot.
facets	Faceting formula to use.
stat	For GRanges: Character vector specifying statistics to use. "stepping" with randomly assigned stepping levels as y varialbe. "identity" allow users to specify y value in aes. For TranscriptDb: defualt "identity" give full gene model and "reduce" for reduced model.
gap.geom	Geom for 'gap' computed from the data you passed based on the group information.
rect.height	Half height of the arrow body.
group.selfish	Passed to addStepping, control whether to show each group as unique level or not. If set to FALSE, if two groups are not overlapped with each other, they will probably be layout in the same level to save space.
which	GRanges object to subset the TranscriptDb object.
xlim	Limits for x, to subset the TranscriptDb object.
truncate.gaps	logical value indicate to truncate gaps or not.
truncate.fun	shrinkage function. Please see shrinkagefun in package biovizBase.
ratio	used in maxGap.
geom	geometric object. only support "gene" now.
range.geom	geom for main intevals or exons.
utr.geom	geom for utr region.
names.expr	Expression for showing y label.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 100
require(GenomicRanges)
## ======================================================================
##  simmulated GRanges
## ======================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))


## ======================================================================
##  default
## ======================================================================
ggplot(gr) + geom_alignment()
## or
ggplot() + geom_alignment(gr)

## ======================================================================
##  facetting and aesthetics
## ======================================================================
ggplot(gr) + geom_alignment(facets = sample ~ seqnames, aes(color = strand, fill = strand))

## ======================================================================
##  stat:stepping
## ======================================================================
ggplot(gr) + geom_alignment(stat = "stepping", aes(group = pair))

## ======================================================================
##  group.selfish controls when
## ======================================================================
ggplot(gr) + geom_alignment(stat = "stepping", aes(group = pair), group.selfish = FALSE)

## ======================================
##  main/gap geom
## ======================================
ggplot(gr) + geom_alignment(range.geom = "arrowrect", gap.geom = "chevron")

## ======================================
##  For TranscriptDb
## ======================================
```

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
## made a track comparing full/reduce stat.
p1 <- ggplot(txdb) + geom_alignment(which = genesymbol["RBM17"])
p2 <- ggplot(txdb) + geom_alignment(which = genesymbol["RBM17"], stat = "reduce")
tracks(full = p1, reduce = p2, heights = c(3, 1))
tracks(full = p1, reduce = p2, heights = c(3, 1)) + theme_tracks_sunset()
tracks(full = p1, reduce = p2, heights = c(3, 1)) +
     theme_tracks_sunset(axis.line.color = NA)
## change y labels
ggplot(txdb) + geom_alignment(which = genesymbol["RBM17"], names.expr = "tx_id:::gene_id")
```

`geom_arch` *Arch geoms for GRanges object*

Description

Show interval data as arches.

Usage

```
## S4 method for signature 'data.frame'
geom_arch(data, ..., n = 25, max.height = 10)


## S4 method for signature 'GRanges'
geom_arch(data, ..., xlab, ylab, main, facets = NULL,
                 rect.height = 0, n = 25, max.height = 10)
```

Arguments

`data`	A `GRanges` or `data.frame` object.
`...`	Extra parameters passed to autoplot function, `aes` mapping support `height, x, xend`. • xstart of the arches • xendend of the arches • heightheight of arches
`xlab`	Label for x
`ylab`	Label for y
`main`	Title for plot.
`n`	Integer values at which interpolation takes place to create 'n' equally spaced points spanning the interval ['min(x)', 'max(x)'].
`facets`	Faceting formula to use.
`rect.height`	When data is `GRanges`, this padding the arches from original y value to allow users putting arches 'around' the interval rectangles.
`max.height`	Max height of all arches.

Details

To draw a interval data as arches, we need to provide a special geom for this purpose. Arches is popular in gene viewer or genomoe browser, when they try to show isoforms or gene model.geom_arch, just like any other geom_* function in ggplot2, you can pass aes() to it to map variable to height of arches.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 100
library(GenomicRanges)

## ======================================
##  simmulated GRanges
## ======================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

## ======================================
##  default
## ======================================
ggplot(gr) + geom_arch()
# or
ggplot() + geom_arch(gr)

## ======================================
##  facetting and aesthetics
## ======================================
ggplot(gr) + geom_arch(aes(color = value, height = value, size = value),
                     alpha = 0.2, facets = sample ~ seqnames)
```

geom_arrow *Arrow geoms for GRanges object*

Description

Show interval data as arrows.

Usage

```
## S4 method for signature 'GRanges'
geom_arrow(data, ..., xlab, ylab, main,
          angle = 30, length = unit(0.15, "cm"), type = "open",
          stat = c("stepping", "identity"), facets = NULL,
          arrow.rate = 0.03, group.selfish = TRUE)
```

Arguments

data	A GRanges object.
...	Extra parameters such as aes() passed.
xlab	Label for x
ylab	Label for y
main	Title for plot.
angle	The angle of the arrow head in degrees (smaller numbers produce narrower, pointier arrows). Essentially describes the width of the arrow head.
length	A unit specifying the length of the arrow head (from tip to base).
type	One of "open" or "closed" indicating whether the arrow head should be a closed triangle.
stat	Character vector specifying statistics to use. "stepping" with randomly assigned stepping levels as y varialbe. "identity" allow users to specify y value in aes.
facets	Faceting formula to use.
arrow.rate	Arrow density of the arrow body.
group.selfish	Passed to addStepping, control whether to show each group as unique level or not. If set to FALSE, if two groups are not overlapped with each other, they will probably be layout in the same level to save space.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 100
require(GenomicRanges)
## ============================================================================
##  simmulated GRanges
## ============================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))


## ============================================================================
##  default
## ============================================================================
ggplot(gr) + geom_arrow()
# or
ggplot() + geom_arrow(gr)

## ============================================================================
##  facetting and aesthetics
## ============================================================================
ggplot(gr) + geom_arrow(facets = sample ~ seqnames, aes(color = strand, fill = strand))

## ============================================================================
##  stat:identity
## ============================================================================
ggplot(gr) + geom_arrow(stat = "identity", aes(y = value))


## ============================================================================
##  stat:stepping
## ============================================================================
ggplot(gr) + geom_arrow(stat = "stepping", aes(y = value, group = pair))

## ============================================================================
##  group.selfish
## ============================================================================
ggplot(gr) + geom_arrow(stat = "stepping", aes(y = value, group = pair), group.selfish = FALSE)
```

```
## ========================================================================
## other options to control arrow angle, density, ...
## ========================================================================
library(grid)
ggplot(gr) + geom_arrow(stat = "stepping", aes(y = value, group = pair),
                      arrow.rate = 0.01, legnth = unit(0.3, "cm"), agnle = 45,
                      group.selfish = FALSE)
```

`geom_arrowrect` *Arrowrect geoms for GRanges object*

Description

Show interval data as rectangle with a arrow head.

Usage

```
## S4 method for signature 'GRanges'
geom_arrowrect(data, ..., xlab, ylab, main,
               facets = NULL, stat = c("stepping", "identity"),
               rect.height = NULL, arrow.head = 0.06,
               arrow.head.rate = arrow.head,  arrow.head.fix = NULL,
               group.selfish = TRUE)
```

Arguments

`data`	A `GRanges` object.
`...`	Extra parameters such as aes() passed.
`xlab`	Label for x
`ylab`	Label for y
`main`	Title for plot.
`facets`	Faceting formula to use.
`stat`	Character vector specifying statistics to use. "stepping" with randomly assigned stepping levels as y varialbe. "identity" allow users to specify y value in `aes`.
`rect.height`	Half height of the arrow body.
`arrow.head`	Arrow head to body ratio.
`arrow.head.rate`	Arrow head to body ratio. same with arrow.head.
`arrow.head.fix`	fixed length of arrow head.
`group.selfish`	Passed to `addStepping`, control whether to show each group as unique level or not. If set to `FALSE`, if two groups are not overlapped with each other, they will probably be layout in the same level to save space.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 100
require(GenomicRanges)
## ============================================================================
##  simmulated GRanges
## ============================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))


## ============================================================================
##  default
## ============================================================================
ggplot(gr) + geom_arrowrect()
## or
ggplot() + geom_arrowrect(gr)

## ============================================================================
##  facetting and aesthetics
## ============================================================================
ggplot(gr) + geom_arrowrect(facets = sample ~ seqnames, aes(color = strand, fill = strand))


## ============================================================================
##  stat:identity
## ============================================================================
ggplot(gr) + geom_arrowrect(stat = "identity", aes(y = value))


## ============================================================================
##  stat:stepping
## ============================================================================
```

```
ggplot(gr) + geom_arrowrect(stat = "stepping", aes(y = value, group = pair))


## ========================================================================
##  group.selfish controls when
## ========================================================================
ggplot(gr) + geom_arrowrect(gr, stat = "stepping", aes(y = value, group = pair), group.selfish = FALSE)
```

---

geom_bar *Segment geoms for GRanges object*

Description

Show interval data as vertical bar, width equals to interval width and use 'score' or specified 'y' as y scale.

Usage

```
##  for data.frame
## S4 method for signature 'data.frame'
geom_bar(data, ...)

## S4 method for signature 'GRanges'
geom_bar(data,..., xlab, ylab, main)
```

Arguments

data	A GRanges or data.frame object.
...	Extra parameters such as aes() or color, size passed.
xlab	Label for x
ylab	Label for y
main	Title for plot.

Details

Useful for showing bed like files, when imported as GRanges, have a extra 'score' column, use it as default y, you could also specify y by using aes(y =).

Value

A 'Layer'.

Examples

```
## load
library(GenomicRanges)

## simul
set.seed(123)
gr.b <- GRanges(seqnames = "chr1", IRanges(start = seq(1, 100, by = 10),
                  width = sample(4:9, size = 10, replace = TRUE)),
                score = rnorm(10, 10, 3), value = runif(10, 1, 100))
gr.b2 <- GRanges(seqnames = "chr2", IRanges(start = seq(1, 100, by = 10),
                  width = sample(4:9, size = 10, replace = TRUE)),
                score = rnorm(10, 10, 3), value = runif(10, 1, 100))
gr.b <- c(gr.b, gr.b2)
## default use score as y

## bar
ggplot(gr.b) + geom_bar(aes(fill = value))
## or
ggplot() + geom_bar(gr.b, aes(fill = value))
ggplot(gr.b) + geom_bar(aes(y = value))
## equal to
autoplot(gr.b, geom = "bar")
```

---

geom_chevron *Chevron geoms for GRanges object*

Description

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

Usage

```
## S4 method for signature 'GRanges'
geom_chevron(data, ..., xlab, ylab, main,
             offset = 0.1,
             facets = NULL,
             stat = c("stepping", "identity"),
             chevron.height.rescale = c(0.1, 0.8),
             group.selfish = TRUE)
```

Arguments

data	A GRanges object.
...	Extra parameters passed to autoplot function.
xlab	Label for x
ylab	Label for y

main	Title for plot.
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.
facets	faceting formula to use.
stat	character vector specifying statistics to use. "stepping" with randomly assigned stepping levels as y varialbe. "identity" allow users to specify y value in aes.
chevron.height.rescale	A numeric vector of length 2. When the offset parameters is a character which is one of the data columns, this parameter rescale the offset.
group.selfish	Passed to addStepping, control whether to show each group as unique level or not. If set to FALSE, if two groups are not overlapped with each other, they will probably be layout in the same level to save space.

Details

To draw a normal GRanges as Chevron, 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

```
set.seed(1)
N <- 100
require(GenomicRanges)

## ======================================================================
##  simmulated GRanges
## ======================================================================
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),
               sample = sample(c("Normal", "Tumor"),
                 size = N, replace = TRUE),
               pair = sample(letters, size = N,
                 replace = TRUE))


## ======================================================================
##  default
##
##  ======================================================================
ggplot(gr) + geom_chevron()
## or
ggplot() + geom_chevron(gr)


## ======================================================================
##  facetting and aesthetics
## ======================================================================
ggplot(gr) + geom_chevron(facets = sample ~ seqnames, aes(color = strand))


## ======================================================================
##  stat:identity
## ======================================================================
ggplot(gr) + geom_chevron(stat = "identity", aes(y = value))


## ======================================================================
##  stat:stepping
## ======================================================================
ggplot(gr) + geom_chevron(stat = "stepping", aes(group = pair))


## ======================================================================
##  group.selfish controls when
## ======================================================================
ggplot(gr) + geom_chevron(stat = "stepping", aes(group = pair), group.selfish = FALSE,
                        xlab = "xlab", ylab = "ylab", main = "main")

p <- qplot(x = mpg, y = cyl, data = mtcars)

## ======================================================================
##  offset
## ======================================================================
gr2 <- GRanges("chr1", IRanges(c(1, 10, 20), width = 5))
gr2.p <- gaps(gr2)
## resize to connect them
gr2.p <- resize(gr2.p, fix = "center", width = width(gr2.p)+2)

ggplot(gr2) + geom_rect() + geom_chevron(gr2.p)
```

```
## notice the rectangle height is 0.8
## offset = 0 just like a line
ggplot(gr2) + geom_rect() + geom_chevron(gr2.p, offset = 0)


## equal height
ggplot(gr2) + geom_rect() + geom_chevron(gr2.p, offset = 0.4)


## ======================================================================
##  chevron.height
## ======================================================================
values(gr2.p)$score <- c(100, 200)
ggplot(gr2) + geom_rect() + geom_chevron(gr2.p, offset = "score")
## chevron.height
ggplot(gr2) + geom_rect() + geom_chevron(gr2.p, offset = "score",
                                         chevron.height.rescale = c(0.4, 10))
```

---

geom_rect *Rect geoms for GRanges object*

Description

Show interval data as rectangle.

Usage

```
## For data.frame
## S4 method for signature 'data.frame'
geom_rect(data, ...)
## For GRanges
## S4 method for signature 'GRanges'
geom_rect(data,..., xlab, ylab, main,
          facets = NULL, stat = c("stepping", "identity"),
          rect.height = NULL,
          group.selfish = TRUE)
```

Arguments

data	A GRanges or data.frame object. When it's data.frame, it's simply calling ggplot2::geom_rect.
...	Extra parameters such as aes() or color, size passed.
xlab	Label for x
ylab	Label for y
main	Title for plot.

facets	Faceting formula to use.
stat	Character vector specifying statistics to use. "stepping" with randomly assigned stepping levels as y varialbe. "identity" allow users to specify y value in aes.
rect.height	Half height of the arrow body.
group.selfish	Passed to addStepping, control whether to show each group as unique level or not. If set to FALSE, if two groups are not overlapped with each other, they will probably be layout in the same level to save space.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 100
require(GenomicRanges)

## ========================================================================
##  simmulated GRanges
## ========================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))



## ========================================================================
##  data.frame call ggplot2::geom_rect
## ========================================================================
ggplot() + geom_rect(data = mtcars, aes(xmin = mpg, ymin = wt, xmax = mpg + 10, ymax = wt + 0.2,
                      fill = cyl))



## ========================================================================
##  default
```

```
## ========================================================================
ggplot(gr) + geom_rect()
# or
ggplot() + geom_rect(gr)


## ========================================================================
##  facetting and aesthetics
## ========================================================================
ggplot(gr) + geom_rect(facets = sample ~ seqnames, aes(color = strand, fill = strand))


## ========================================================================
##  stat:identity
## ========================================================================
ggplot(gr) + geom_rect(stat = "identity", aes(y = value))


## ========================================================================
##  stat:stepping
## ========================================================================
ggplot(gr) + geom_rect(stat = "stepping", aes(y = value, group = pair))


## ========================================================================
##  group.selfish controls when
## ========================================================================
ggplot(gr) + geom_rect(stat = "stepping", aes(y = value, group = pair), group.selfish = FALSE)
```

---

geom_segment *Segment geoms for GRanges object*

Description

Show interval data as segments.

Usage

```
##  for data.frame
## S4 method for signature 'data.frame'
geom_segment(data, ...)

## S4 method for signature 'GRanges'
geom_segment(data,..., xlab, ylab, main,
          facets = NULL, stat = c("stepping", "identity"),
          group.selfish = TRUE)
```

Arguments

data	A GRanges or data.frame object.
...	Extra parameters such as aes() or color, size passed.
xlab	Label for x
ylab	Label for y
main	Title for plot.
facets	Faceting formula to use.
stat	Character vector specifying statistics to use. "stepping" with randomly assigned stepping levels as y varialbe. "identity" allow users to specify y value in aes.
group.selfish	Passed to addStepping, control whether to show each group as unique level or not. If set to FALSE, if two groups are not overlapped with each other, they will probably be layout in the same level to save space.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 100
require(GenomicRanges)

## ======================================================================
##  simmulated GRanges
## ======================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))


## ======================================================================
##  data.frame call ggplot2::geom_segment
## ======================================================================
```

```
ggplot() + geom_segment(data = mtcars, aes(x = mpg, y = wt, xend = mpg + 10, yend = wt + 0.2,
                        fill = cyl))


## ======================================================================
##  default
##
##  ======================================================================
ggplot(gr) + geom_segment()
## or
ggplot() + geom_segment(gr)


## ======================================================================
##  facetting and aesthetics
## ======================================================================
ggplot(gr) + geom_segment(facets = sample ~ seqnames, aes(color = strand, fill = strand))


## ======================================================================
##  stat:identity
## ======================================================================
ggplot(gr) + geom_segment(stat = "identity", aes(y = value))


## ======================================================================
##  stat:stepping
## ======================================================================
ggplot(gr) + geom_segment(stat = "stepping", aes(y = value, group = pair))


## ======================================================================
##  group.selfish controls when
## ======================================================================
ggplot(gr) + geom_segment(stat = "stepping", aes(y = value, group = pair), group.selfish = FALSE)
```

---

ggbio *class ggbio*

Description

a sub class of ggplot and gg class defined in ggplot2 package, used for ggbio specific methods.

Usage

```
ggbio(x)
```

Arguments

x	a ggplot or gg object.

Details

This class is defined to facilitate the ggbio-specific visualization method, especially when using ggplot to construct ggbio supported object, that will return a ggbio class. And internals tricks will help a lazy evaluation for following + method.

Value

a ggbio object.

Author(s)

Tengfei Yin

See Also

ggplot

Examples

```
p1 <- qplot()
g1 <- ggbio(p1)
class(g1)
```

---

ggplot *ggplot generic method*

Description

Usage is similar to ggplot function in ggplot2 package, but support more objects defined in Bioconductor, or even for object like matrix, which intitialize a ggbio object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden. And more than that, it store the original data object before coercion into the returned object. Please check the behavior for mold method to see how each object coerced into data.frame.

Usage

```
## S4 method for signature 'GRanges'
ggplot(data, ...)
## S4 method for signature 'GRangesList'
ggplot(data, ...)
## S4 method for signature 'IRanges'
ggplot(data, ...)
## S4 method for signature 'Seqinfo'
ggplot(data, ...)
## S4 method for signature 'matrix'
ggplot(data, ...)
```

```
## S4 method for signature 'Views'
ggplot(data, ...)
## S4 method for signature 'ExpressionSet'
ggplot(data, ...)
## S4 method for signature 'SummarizedExperiment'
ggplot(data, assay.id = 1, ...)
## S4 method for signature 'VCF'
ggplot(data, ...)
## S4 method for signature 'GappedAlignments'
ggplot(data, ...)
## S4 method for signature 'BamFile'
ggplot(data, ...)
## S4 method for signature 'character'
ggplot(data, ...)
## S4 method for signature 'TranscriptDb'
ggplot(data, ...)
## S4 method for signature 'BSgenome'
ggplot(data, ...)
## S4 method for signature 'Rle'
ggplot(data, ...)
## S4 method for signature 'RleList'
ggplot(data, ...)
```

Arguments

data	original data object.
...	other arguments passed to specific methods.
assay.id	index of assay you are using when multiple assays exist.

Details

The biggest difference for objects returned by ggplot in ggbio from ggplot2, is we always keep the original data copy, this is useful because in ggbio, our starting point is not always data.frame, many special statistical transformation is computed upon original data objects instead of coerced data.frame. This is a hack to follow ggplot2's API while allow our own defined components to trace back to original data copy and do the transformation. For objects supported by mold we transform them to data.frame stored along the original data set, for objects which not supported by mold method, we only store the original copy for ggbio specific graphics.

ggplot() is typically used to construct a plot incrementally, using the + operator to add layers to the existing ggplot object. This is advantageous in that the code is explicit about which layers are added and the order in which they are added. For complex graphics with multiple layers, initialization with ggplot is recommended. You can always call qplot in package ggplot2 or autoplot in ggbio for convenient usage.

There are three common ways to invoke ggplot:

- ggplot(df, aes(x, y, <other aesthetics>))
- ggplot(df)
- ggplot()

The first method is recommended if all layers use the same data and the same set of aesthetics, although this method can also be used to add a layer using data from another data frame. The second method specifies the default data frame to use for the plot, but no aesthetics are defined up front. This is useful when one data frame is used predominantly as layers are added, but the aesthetics may vary from one layer to another. The third method initializes a skeleton ggplot object which is fleshed out as layers are added. This method is useful when multiple data frames are used to produce different layers, as is often the case in complex graphics.

The examples below illustrate how these methods of invoking ggplot can be used in constructing a graphic.

Value

a return ggbio object, which is a subclass of ggplot defined in ggplot2 package, but that's more, a '.data' list entry is stored with the returned object.

Author(s)

Tengfei Yin

See Also

mold, ggbio

Examples

```
set.seed(1)
N <- 100
library(GenomicRanges)
## GRanges
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

## automatically facetting and assign y
## this must mean geom_rect support GRanges object
ggplot(gr) + geom_rect()
ggplot(gr) + geom_alignment()
ggplot() + geom_alignment(gr)


## use pure ggplot2's geom_rect, no auto facet
```

```
ggplot(gr) + ggplot2::geom_rect(aes(xmin = start, ymin = score,
                                xmax = end, ymax = score + 1))

## GRangesList
grl <- split(gr, values(gr)$pair)
ggplot(grl) + geom_alignment()
ggplot(grl) + geom_rect()
ggplot(grl) + ggplot2::geom_rect(aes(xmin = start, ymin = score,
                                xmax = end, ymax = score + 1))


## IRanges
ir <- ranges(gr)
ggplot(ir) + geom_rect()
ggplot(ir) + layout_circle(geom = "rect")

## Seqinfo
seqlengths(gr) <- c(400, 500, 420)
ggplot(seqinfo(gr)) + geom_point(aes(x = midpoint, y = seqlengths))

## matrix
mx <- matrix(1:12, nrow = 3)
ggplot(mx, aes(x = x, y = y)) + geom_raster(aes(fill = value))
## row is the factor
ggplot(mx, aes(x = x, y = row)) + geom_raster(aes(fill = value))
colnames(mx)
colnames(mx) <- letters[1:ncol(mx)]
mx
## has extra 'colnames'
ggplot(mx, aes(x = x, y = row)) + geom_raster(aes(fill = colnames))
rownames(mx)
rownames(mx) <- LETTERS[1:nrow(mx)]
ggplot(mx, aes(x = x, y = row)) + geom_raster(aes(fill = rownames))
## please check autoplot, matrix for more control

##  Views









## ExpressionSet
library(Biobase)
data(sample.ExpressionSet)
sample.ExpressionSet
set.seed(1)
## select 50 features
idx <- sample(seq_len(dim(sample.ExpressionSet)[1]), size = 50)
```

```
eset <- sample.ExpressionSet[idx,]

ggplot(eset) + geom_tile(aes(x = x, y = y, fill = value))
## please check autoplot,matrix method which gives you more control
head(ggplot(eset)$data)
ggplot(eset) + geom_tile(aes(x = x, y = y, fill = sex))
ggplot(eset) + geom_tile(aes(x = x, y = y, fill = type))

## Rle
library(IRanges)
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)
ggplot(xRle) + geom_tile(aes(x = x, y = y, fill = value))

## RleList
xRleList <- RleList(xRle, 2L * xRle)
xRleList
ggplot(xRleList) + geom_tile(aes(x = x, y = y, fill = value)) +
facet_grid(group~.)
names(xRleList) <- c("a" ,"b")
ggplot(xRleList) + geom_tile(aes(x = x, y = y, fill = value)) +
facet_grid(group~.)


## SummarizedExperiments
library(GenomicRanges)
nrows <- 200; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
counts2 <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowData <- GRanges(rep(c("chr1", "chr2"), c(50, 150)),
                   IRanges(floor(runif(200, 1e5, 1e6)), width=100),
                   strand=sample(c("+", "-"), 200, TRUE))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                     row.names=LETTERS[1:6])
sset <- SummarizedExperiment(assays=SimpleList(counts=counts,
                                               counts2 = counts2),
                             rowData=rowData, colData=colData)
ggplot(sset) + geom_raster(aes(x = x, y = y , fill = value))
```

---

ggsave *Save a ggplot object or tracks with sensible defaults*

Description

ggsave is a convenient function for saving a plot. It defaults to saving the last plot that you displayed, and for a default size uses the size of the current graphics device. It also guesses the type of graphics device from the extension. This means the only argument you need to supply is the filename.

Usage

```
ggsave(filename = default_name(plot), plot = last_plot(),
  device = default_device(filename), path = NULL,
  scale = 1, width = par("din")[1],
  height = par("din")[2], units = c("in", "cm", "mm"),
  dpi = 300, limitsize = TRUE, ...)
```

Arguments

filename	file name/filename of plot
plot	plot to save, defaults to last plot displayed
device	device to use, automatically extract from file name extension
path	path to save plot to (if you just want to set path and not filename)
scale	scaling factor
width	width (defaults to the width of current plotting window)
height	height (defaults to the height of current plotting window)
units	units for width and height when either one is explicitly specified (in, cm, or mm)
dpi	dpi to use for raster graphics
limitsize	when TRUE (the default), ggsave will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.
...	other arguments passed to graphics device

Details

ggsave currently recognises the extensions eps/ps, tex (pictex), pdf, jpeg, tiff, png, bmp, svg and wmf (windows only).

---

layout_circle *Create a circle layout*

Description

Create a circle layout.

Usage

```
## S4 method for signature 'GRanges'
layout_circle(data, ..., geom = c("point", "line", "link", "ribbon",
               "rect", "bar", "segment", "hist", "scale", "heatmap", "ideogram",
                 "text"), linked.to, radius = 10, trackWidth = 5,
                 space.skip = 0.015, direction = c("clockwise",
                 "anticlockwise"), link.fun = function(x, y, n = 30)
                 bezier(x, y, evaluation = n), rect.inter.n = 60, rank,
                 ylim = NULL,
```

```
                  scale.n = 60, scale.unit = NULL, scale.type = c("M",
                  "B", "sci"), grid.n = 5, grid.background = "gray70",
                  grid.line = "white", grid = FALSE, chr.weight = NULL)

## S4 method for signature 'missing'
layout_circle(data, ...)
```

Arguments

data	A GRanges object.
...	Extra parameters such as aesthetics mapping in aes(), or color, size, etc.
geom	The geometric object to use display the data.
linked.to	Character indicates column that specifying end of the linking lines, that column should be a GRanges object.
radius	Numeric value indicates radius. Default is 10.
trackWidth	Numeric value indicates the track width.
space.skip	Numeric value indicates the ratio of skipped region between chunks(chromosomes in GRanges) to the whole track space.
direction	Space layout orders.
link.fun	Function used for interpolate the linking lines. Default is Hmisc::bezier.
rect.inter.n	n passed to interpolate function in rectangle transformation(from a rectangle) to a section in circular view.
rank	For default equal trackWidth, use rank to specify the circle orders.
ylim	Numeric range to control y limits.
scale.n	Approximate number of ticks you want to show on the whole space. used when scale.unit is NULL.
scale.unit	Unit used for computing scale. Default is NULL,
scale.type	Scale type used for
grid	logical value indicate showing grid background for track or not.
grid.n	integer value indicate horizontal grid line number.
grid.background	grid background color.
grid.line	grid line color.
chr.weight	numeric vectors which sum to <1, the names of vectors has to be matched with seqnames in seqinfo, and you can only specify part of the seqnames, other lengths of chromosomes will be assined proportionally to their seqlengths, for example, you could specify chr1 to be 0.5, so the chr1 will take half of the space and other chromosomes squeezed to take left of the space.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
N <- 100
library(GenomicRanges)
## ============================================================================
##  simmulated GRanges
## ============================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))


seqlengths(gr) <- c(400, 500, 700)
values(gr)$to.gr <- gr[sample(1:length(gr), size = length(gr))]

## doesn't pass gr to the ggplot
ggplot() + layout_circle(gr, geom = "ideo", fill = "gray70", radius = 7, trackWidth = 3) +
 layout_circle(gr, geom = "bar", radius = 10, trackWidth = 4, aes(fill = score, y = score)) +
  layout_circle(gr, geom = "point", color = "red", radius = 14,
                trackWidth = 3, grid = TRUE, aes(y = score)) +
    layout_circle(gr, geom = "link", linked.to = "to.gr", radius = 6,
trackWidth = 1)

## more formal API
ggplot(gr) + layout_circle(geom = "ideo", fill = "gray70", radius = 7, trackWidth = 3) +
  layout_circle(geom = "bar", radius = 10, trackWidth = 4, aes(fill = score, y = score)) +
  layout_circle(geom = "point", color = "red", radius = 14,
                trackWidth = 3, grid = TRUE, aes(y = score)) +
    layout_circle(geom = "link", linked.to = "to.gr", radius = 6, trackWidth = 1)
```

---

layout_karyogram *Create a karyogram layout*

Description

Create a karyogram layout.

Usage

```
## S4 method for signature 'GRanges'
layout_karyogram(data, ..., xlab, ylab, main,
          facets = seqnames ~ ., cytoband = FALSE, geom = "rect",
          stat = NULL, ylim = NULL, rect.height = 10)
```

Arguments

data	a GRanges object, which could contain extra information about cytoband. If you want an accurate genome mapping, please provide seqlengths with this GRanges object,otherwise it will emit a warning and use data space to estimate the chromosome space which is very rough.
...	Extra parameters such as aes() or arbitrary color and size.
xlab	character vector or expression for x axis label.
ylab	character vector or expression for y axis label.
main	character vector or expression for plot title.
facets	faceting formula to use.
cytoband	logical value indicate to show the cytobands or not.
geom	The geometric object to use display the data.
stat	character vector specifying statistics to use.
ylim	limits for y axis, usually the chromosome spaces y limits are from 0 to rect.height, which 10, so if you wan to stack some data on top of it, you can set limits to like c(10, 20).
rect.height	numreic value indicate half of the rectangle ploting region, used for alignment of multiple layers.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
### R code from vignette source 'karyogram.Rnw'

###################################################
### code chunk number 1: loading
###################################################
library(ggbio)
data(hg19IdeogramCyto, package = "biovizBase")
head(hg19IdeogramCyto)
## default pre-set color stored in
```

```
getOption("biovizBase")$cytobandColor


####################################################
### code chunk number 2: default
####################################################
autoplot(hg19IdeogramCyto, layout = "karyogram", cytoband = TRUE)



####################################################
### code chunk number 3: change-order
####################################################
library(GenomicRanges)
hg19 <- keepSeqlevels(hg19IdeogramCyto, paste0("chr", c(1:22, "X", "Y")))
head(hg19)
autoplot(hg19, layout = "karyogram", cytoband = TRUE)


####################################################
### code chunk number 4: cyto-normal
####################################################
library(GenomicRanges)
## it's a 'ideogram'
biovizBase::isIdeogram(hg19)
## set to FALSE
autoplot(hg19, layout = "karyogram", cytoband = FALSE, aes(fill = gieStain)) +
  scale_fill_giemsa()


####################################################
### code chunk number 5: load-RNAediting
####################################################
data(darned_hg19_subset500, package = "biovizBase")
dn <- darned_hg19_subset500
head(dn)
## add seqlengths
## we have seqlegnths information in another data set
data(hg19Ideogram, package = "biovizBase")
seqlengths(dn) <- seqlengths(hg19Ideogram)[names(seqlengths(dn))]
## now we have seqlengths
head(dn)
## then we change order
dn <- keepSeqlevels(dn, paste0("chr", c(1:22, "X")))
autoplot(dn, layout = "karyogram")
## this equivalent to
## autoplot(seqinfo(dn))


####################################################
### code chunk number 6: load-RNAediting-color
####################################################
## since default is geom rectangle, even though it's looks like segment
```

```
## we still use both fill/color to map colors
autoplot(dn, layout = "karyogram", aes(color = exReg, fill = exReg))


#####################################################
### code chunk number 7: load-RNAediting-color-NA
#####################################################
## since default is geom rectangle, even though it's looks like segment
## we still use both fill/color to map colors
autoplot(dn, layout = "karyogram", aes(color = exReg, fill = exReg)) +
  scale_color_discrete(na.value = "brown")


#####################################################
### code chunk number 8: load-RNAediting-color-fake
#####################################################
dn2 <- dn
seqlengths(dn2) <- rep(max(seqlengths(dn2)), length(seqlengths(dn2)) )
autoplot(dn2, layout = "karyogram", aes(color = exReg, fill = exReg))


#####################################################
### code chunk number 9: plotKaryogram (eval = FALSE)
#####################################################
## plotKaryogram(dn)
## plotKaryogram(dn, aes(color = exReg, fill = exReg))


#####################################################
### code chunk number 10: low-default
#####################################################
## plot ideogram
p <- ggplot(hg19) + layout_karyogram(cytoband = TRUE)
p
## eqevelant autoplot(hg19, layout = "karyogram", cytoband = TRUE)


#####################################################
### code chunk number 11: low-default-addon
#####################################################
p <- p + layout_karyogram(dn, geom = "rect", ylim = c(11, 21), color = "red")
## commented line below won't work
## the cytoband fill color has been used already.
## p <- p + layout_karyogram(dn, aes(fill = exReg, color = exReg), geom = "rect")
p


#####################################################
### code chunk number 12: edit-space
#####################################################
## plot chromosome space
p <- autoplot(seqinfo(dn))
## make sure you pass rect as geom
```

```
## otherwise you just get background
p <- p + layout_karyogram(dn, aes(fill = exReg, color = exReg), geom = "rect")
values(dn)$pvalue <- rnorm(length(dn))
p + layout_karyogram(dn, aes(x = start, y = pvalue), ylim = c(10, 30), geom = "line", color = "red")
p


################################################
### code chunk number 13: sessionInfo
################################################
sessionInfo()
```

---

mold *Mold an object to a data frame*

Description

S4 method to transform a object to a data.frame.

Usage

```
## S4 method for signature 'eSet'
mold(data)
## S4 method for signature 'GRanges'
mold(data)
## S4 method for signature 'IRanges'
mold(data)
## S4 method for signature 'GRangesList'
mold(data,
          indName = "grl_name")
## S4 method for signature 'IRanges'
mold(data)
## S4 method for signature 'Seqinfo'
mold(data)
## S4 method for signature 'matrix'
mold(data)
## S4 method for signature 'ExpressionSet'
mold(data)
## S4 method for signature 'SummarizedExperiment'
mold(data, assay.id = 1)
## S4 method for signature 'Views'
mold(data)
## S4 method for signature 'Rle'
mold(data)
## S4 method for signature 'RleList'
mold(data)
```

Arguments

data	oringinal data object.
indName	character. When mold a GRangesList to a GRanges the collapsed list names will be added as a column named by indName, default is "grl_name".
assay.id	an integer indicates which assay to be used to mold into the data.frame.

Details

For different object, we try to maximize the information kept during molding to a data.frame. Most cases, this is different from simply use method as.data.frame.

return a data.frame with extra 'midpoint' column, which is (start+end)/2.

GRangesGRangesList return a data.frame with extra 'midpoint' column, which is (start+end)/2, and with indName indicates which group they are originally from the list.

IRanges return a data.frame with extra 'midpoint' column, which is (start+end)/2.

Seqinfo return a data.frame column: seqnames, start, end, width, strand, midpoint, seqlengths, isCircular, genome.

matrix return a data.frame with 'x', 'y', and 'value', 'row', 'col' column. If either colnames or rownames exists, a new 'colnames' or 'rownames' column will be created and added to the data.frame. Notice, 'x' and 'y' are numeric coordinates in the matrix while 'col' and 'row' are the same value but are all factors.

Views return a data.frame with 'x', 'y', and 'value', 'start', 'end', 'width', 'midpoint', 'group' column. If either colnames or rownames exists, a new 'colnames' or 'rownames' column will be created and added to the data.frame. This is achieved by coerce it to a matrix first. Additional variable 'row' will be added to indicate the group, but it actually equals to 'y'.

ExpressionSet parse the matrix by using exprs on it and then mold the matrix to a data.frame with 'x', 'y', and 'value', 'col', 'row' column, 'colnames' for sample data and 'rownames' for features. 'x' and 'y' are numeric coordinates in the matrix while 'col' and 'row' are the same value but are all factors. The pheno data is also integrated with it.

Rle coerce to a data.frame with column 'x', 'y', 'col', 'row', 'value'. 'x' and 'y' are numeric coordinates in the matrix while 'col' and 'row' are the same value but are all factors.

RleList coerce to a data.frame with column 'x', 'y','col', 'row', 'value' and 'group', and 'group' variable indicates the original list entry number. 'x' and 'y' are numeric coordinates in the matrix while 'col' and 'row' are the same value but are all factors.

SummarizedExperiment parse the matrix by using exprs on it and then mold the matrix to a data.frame with 'x', 'y', and 'value', 'col', 'row' column, 'colnames' for sample data and 'rownames' for features. 'x' and 'y' are numeric coordinates in the matrix while 'col' and 'row' are the same value but are all factors. The colData and rowData are also integrated with it.

Value

a data.frame object.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 1000
library(GenomicRanges)
## GRanges
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

## GRangesList
grl <- split(gr, values(gr)$pair)
head(mold(grl))
head(mold(grl, indName = "group_sample"))

## IRanges
ir <- ranges(gr)
head(mold(ir))

## Seqinfo
seqlengths(gr) <- c(400, 500, 420)
head(mold(seqinfo(gr)))


## matrix
mx <- matrix(1:12, nrow = 3)
head(mold(mx))
colnames(mx)
colnames(mx) <- letters[1:ncol(mx)]
mx
head(mold(mx))
rownames(mx)
rownames(mx) <- LETTERS[1:nrow(mx)]
head(mold(mx))
```

```
## ExpressionSet
library(Biobase)
data(sample.ExpressionSet)
sample.ExpressionSet
set.seed(1)
## select 50 features
idx <- sample(seq_len(dim(sample.ExpressionSet)[1]), size = 50)
eset <- sample.ExpressionSet[idx,]
head(mold(eset))



## Rle
library(IRanges)
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)
head(mold(xRle))


## RleList
xRleList <- RleList(xRle, 2L * xRle)
xRleList
head(mold(xRleList))
names(xRleList) <- c("a" ,"b")
xRleList
head(mold(xRleList))

## SummerizedExperiments
library(GenomicRanges)
nrows <- 200; ncols <- 6
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
counts2 <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
rowData <- GRanges(rep(c("chr1", "chr2"), c(50, 150)),
                   IRanges(floor(runif(200, 1e5, 1e6)), width=100),
                   strand=sample(c("+", "-"), 200, TRUE))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                     row.names=LETTERS[1:6])
sset <- SummarizedExperiment(assays=SimpleList(counts=counts,
                                               counts2 = counts2),
                             rowData=rowData, colData=colData)
head(mold(sset))

## VCF
library(VariantAnnotation)
vcffile <- system.file("extdata", "chr22.vcf.gz", package="VariantAnnotation")
vcf <- readVcf(vcffile, "hg19")
```

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 <- TxDb.Hsapiens.UCSC.hg19.knownGene
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, ..., facets, space.skip = 0.01, geom = NULL,
                cutoff = NULL, cutoff.color = "red", cutoff.size = 1,
                legend = FALSE, xlim, ylim, xlab, ylab, main,
                highlight.gr = NULL, highlight.name = NULL,
                highlight.col = "red", highlight.label = TRUE,
                highlight.label.size = 5, highlight.label.offset =
                0.05, highlight.label.col = "black")
```

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).
...	extra arguments passed. such as color, size, alpha.
facets	facets formula, such as group ~ .
space.skip	numeric value for skip ratio, between chromosome spaces.default is 0.01.

geom	geometric object, defualt is "point".
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.
xlim	limits for x scale.
ylim	limits for y scale.
xlab	Label for xscale.
ylab	Label for yscale.
main	title.
highlight.gr	a GRanges object, this wil highlight overlapped region with provided intervals.
highlight.name	if NULL, using rownames of GRanges object provided by argument highlight.gr, otherwise use character to indicate column used as labeled names.
highlight.col	highlight colors.
highlight.label	logical value, label the highlighted region of not.
highlight.label.size	highlight label size.
highlight.label.offset	highlight label offset.
highlight.label.col	highlight label color.

Details

Please use seqlengths of the object and space.skip arguments to control the layout of the coordiant genome transformation.

aes(y = ...) is requried.

aes(color =) is used to mapping to data variables, if just pass "color" without aes(), then will recycle the color to represent each chromosomes.please see the example below.

Value

Return a ggplot object.

Author(s)

Tengfei Yin

Examples

```
##  load
library(ggbio)
data(hg19IdeogramCyto, package = "biovizBase")
data(hg19Ideogram, package = "biovizBase")
library(GenomicRanges)

##  simul_gr
library(biovizBase)
gr <- GRanges(rep(c("chr1", "chr2"), each = 5),
              IRanges(start = rep(seq(1, 100, length = 5), times = 2),
                      width = 50))
autoplot(gr)

##  coord:genome
autoplot(gr, coord = "genome")
gr.t <- transformToGenome(gr)
head(gr.t)

##  is
is_coord_genome(gr.t)
metadata(gr.t)$coord


##  simul_snp
chrs <- as.character(levels(seqnames(hg19IdeogramCyto)))
seqlths <- seqlengths(hg19Ideogram)[chrs]
set.seed(1)
nchr <- length(chrs)
nsnps <- 100
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)
                  )

##  shorter
seqlengths(gr.snp)
nms <- seqnames(seqinfo(gr.snp))
nms.new <- gsub("chr", "", nms)
names(nms.new) <- nms
gr.snp <- renameSeqlevels(gr.snp, nms.new)
seqlengths(gr.snp)
```

```
##  unorder
autoplot(gr.snp, coord = "genome", geom = "point", aes(y = pvalue), space.skip = 0.01)

##  sort
gr.snp <- keepSeqlevels(gr.snp, c(1:22, "X", "Y"))
autoplot(gr.snp, coord = "genome", geom = "point", aes(y = pvalue), space.skip = 0.01)

##  with_seql
names(seqlths) <- gsub("chr", "", names(seqlths))
seqlengths(gr.snp) <- seqlths[names(seqlengths(gr.snp))]
autoplot(gr.snp, coord = "genome", geom = "point", aes(y = pvalue), space.skip = 0.01)

##  line
autoplot(gr.snp, coord = "genome", geom = "line", aes(y = pvalue, group = seqnames,
                                    color = seqnames))



##  plotGrandLinear
plotGrandLinear(gr.snp, aes(y = pvalue))

##  morecolor
plotGrandLinear(gr.snp, aes(y = pvalue, color = seqnames))
plotGrandLinear(gr.snp, aes(y = pvalue), color = c("green", "deepskyblue"))
plotGrandLinear(gr.snp, aes(y = pvalue), color = c("green", "deepskyblue", "red"))
plotGrandLinear(gr.snp, aes(y = pvalue), color = "red")

##  cutoff
plotGrandLinear(gr.snp, aes(y = pvalue), cutoff = 3, cutoff.color = "blue", cutoff.size = 4)

##  cutoff-low
plotGrandLinear(gr.snp, aes(y = pvalue)) + geom_hline(yintercept = 3, color = "blue", size = 4)

##  longer
## let's make a long name
nms <- seqnames(seqinfo(gr.snp))
nms.new <- paste("chr00000", nms, sep = "")
names(nms.new) <- nms
gr.snp <- renameSeqlevels(gr.snp, nms.new)
seqlengths(gr.snp)

##  rotate
plotGrandLinear(gr.snp, aes(y = pvalue)) + theme(axis.text.x=element_text(angle=-90, hjust=0))

##  sessionInfo
sessionInfo()
```

---

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,
                                    stat.ylab,
                                    sig, sig.col = c("black", "red"),
                                    stat.coord.trans = coord_trans(),
                                    annotation = list(),
                                    width.ratio = 0.8,
                                    theme.stat = theme_grey(),
                                    theme.align = theme_gray(),
                                    linetype = 3,
                                    heights)
```

Arguments

`data`	GRanges object with a DataFrame as elementMetadata.
`...`	Parameters passed to control lines in top plot.
`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.
`stat.ylab`	y label for stat track(the top track).
`sig`	a character of element meta data column of logical value, indicates which row is signficant. and will be shown in link lines and rectangle.
`sig.col`	colors for significant, valid when you specify "sig" argument, the first color indicates `FALSE`, non-significant, the second color indicates `TRUE`.
`stat.coord.trans`	transformation used for top plot.
`annotation`	A list of ggplot object.
`width.ratio`	Control the segment length of statistic layer.
`theme.stat`	top plot theme.
`theme.align`	alignment themes.
`linetype`	linetype
`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

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(ggbio)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
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)$score <- rnorm(length(exon.new))
values(exon.new)$significant <- sample(c(TRUE,FALSE), size = length(exon.new),replace = TRUE)

plotRangesLinkedToData(exon.new, stat.col = c("sample1", "sample2"))
plotRangesLinkedToData(exon.new, stat.col = 1:2)
plotRangesLinkedToData(exon.new, stat.col = 1:2, size = 3, linetype = 4)
plotRangesLinkedToData(exon.new, stat.col = 1:2, size = 3, linetype = 4,
                       sig = "significant")
plotRangesLinkedToData(exon.new, stat.col = 1:2, size = 3, linetype = 4,
                        sig = "significant", sig.col = c("gray90","red"))
```

plotSingleChrom *Plot single chromosome with cytoband*

Description

Plot single chromosome with cytoband. `plotIdeogram` is synonyms to `plotSingleChrom`.

Usage

```
plotSingleChrom(obj, subchr, zoom.region, xlab, ylab, main, xlabel =
                FALSE, color = "red", fill = "red", alpha = 0.7,
                zoom.offset = 0.1, size = 1,
                cytoband = TRUE, aspect.ratio = 1/20, genome)

plotIdeogram(obj, subchr, zoom.region, xlab, ylab, main, xlabel =
                FALSE, color = "red", fill = "red", alpha = 0.7,
                zoom.offset = 0.1, size = 1,
                cytoband = TRUE, aspect.ratio = 1/20, genome)
```

Arguments

`obj`	A `GenomicRanges` object, which include extra information about cytoband, check biovizBase::isIdeogram.
`subchr`	A single character of chromosome names to show.
`zoom.region`	A numeric vector of length 2 indicating zoomed region.
`xlab`	Label for x
`ylab`	Label for y
`main`	Title for plot.
`xlabel`	A logical value. Show the x label or not.
`color`	color for highlight region.
`fill`	fill color for highlight region.
`alpha`	alpha for highlight regio.
`zoom.offset`	zoomed highlights region offset around chromosome plotting region.
`size`	size for zoomed region rectangle boundary.
`cytoband`	If FALSE, plot just blank chromosome without cytoband. default is TRUE. es
`aspect.ratio`	aspect ratio for the chromosome ideogram plot, default is NULL.
`genome`	genome character passed to `getIdeogram`

Details

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

Value

A ggplot object.

Author(s)

Tengfei Yin

Examples

```
## Not run:
library(biovizBase)
data(hg19IdeogramCyto, package = "biovizBase")
biovizBase::isIdeogram(hg19IdeogramCyto) ## return TRUE
plotIdeogram(hg19IdeogramCyto, "chr1")
plotIdeogram(hg19IdeogramCyto, "chr1", xlabel = TRUE)
## zoom
plotIdeogram(hg19IdeogramCyto, "chr1",  zoom.region = c(1e8, 1.5e8))

## End(Not run)
```

---

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 <- TxDb.Hsapiens.UCSC.hg19.knownGene
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)
```

---

plotStackedOverview *Plot stacked overview*

Description

Plot stacked overview for genome with or without cytoband. It's a wrapper around layout_karyogram.

Usage

```
plotStackedOverview(obj, ..., xlab, ylab, main, geom = "rect",
                        cytoband = FALSE, rescale = TRUE,
                        rescale.range = c(0, 10))
plotKaryogram(obj, ..., xlab, ylab, main, geom = "rect",
                        cytoband = FALSE, rescale = TRUE,
                        rescale.range = c(0, 10))
```

Arguments

obj	a GRanges object, which could contain extra information about cytoband. If it's missing, will ask user to provide species information and download proper data set from UCSC. If you want an accurate genome mapping, please provide seqlengths with this GRanges object,otherwise it will emit a warning and use data space to estimate the chromosome space which is very rough.
...	arguments passed to graphic functions to control aesthetics. For example, if you use geom "point", you need to provide "y" in aes(), and if can also pass color, fill, size etc. to control graphics.
xlab	label for x
ylab	label for y
main	title for plot.
geom	geom plotted on the stacked layout. Default is "rect", which showing interval data as rectangles. It automatically figures out boundary so you don't have to provide information in aes, users could specify other supported geom works for data.frame.
cytoband	logical value. Default is FALSE. If TRUE, plotting cytoband, this require your data have arbitrary column as name and gieStain. the easiest way is to use getIdeogram to get your data. Notice for this function, when cytoband is TRUE, it will only plot cytoband without overlaying your data. If you really need to overlay extra data on cytoband, please plus layout_karyogram for that purpose.
rescale	logical value. Default is TRUE, which rescale your data into the rescale.range, this make sure your data will not be plotted outside the stacked overview box.
rescale.range	Numeric range of length 2. Default is (0, 10), because stacked layout draws a white background as chromosome space and this space is of height 10. We hide the y-axis since we don't need it for stacked overview. Sometime users may want to leave some margin for their data, they can use this arguments to control the rescale.

Details

Stacked overview is just a arbitrary layout for karyogram layout, it use facets seqnaems ~ . as default to stack the genome. For accurate mapping, you need to provide seqlengths information in your GRanges object. Otherwise, data space will be computed for stacked overview chromosome background, this is _NOT_ the actual chromosome space!.

Value

A ggplot object.

Author(s)

Tengfei Yin

Examples

```
## Not run:
library(biovizBase)
data(hg19IdeogramCyto, package = "biovizBase")
library(GenomicRanges)

## you can also get ideogram by biovizBase::getIdeogram

## make shorter and clean labels
old.chrs <- seqnames(seqinfo(hg19IdeogramCyto))
new.chrs <- gsub("chr", "", old.chrs)
## lst <- as.list(new.chrs)
names(new.chrs) <- old.chrs
new.ideo <- renameSeqlevels(hg19IdeogramCyto, new.chrs)
new.ideo <- keepSeqlevels(new.ideo, c(as.character(1:22) , "X", "Y"))
new.ideo


## sample data
data(darned_hg19_subset500, package = "biovizBase")
idx <- is.na(values(darned_hg19_subset500)$exReg)
values(darned_hg19_subset500)$exReg[idx] <- "unknown"

## you need to add seqlengths for accruate mapping
chrnames <- unique(as.character(seqnames(darned_hg19_subset500)))
data(hg19Ideogram, package = "biovizBase")
seqlengths(darned_hg19_subset500) <- seqlengths(hg19Ideogram)[sort(chrnames)]


dn <- darned_hg19_subset500
values(dn)$score <- rnorm(length(dn))

## plotStackedOverview is a simple wrapper around this functions to
   create a stacked layout
plotStackedOverview(new.ideo, cytoband = TRUE)

plotStackedOverview(dn)
plotStackedOverview(dn, aes(color = exReg, fill = exReg))
## this will did the trick for you to rescale the space
plotStackedOverview(dn, aes(x = midpoint, y = score), geom = "line")
plotStackedOverview(dn, aes(x = midpoint, y = score), geom = "line", rescale.range = c(4, 6))
## no rescale
plotStackedOverview(dn, aes(x = midpoint, y = score), geom = "line", rescale = FALSE,
                    xlab = "xlab", ylab = "ylab", main  = "main") + ylab("ylab")
```

```
## no object? will ask you for species and query the data on the fly
plotStackedOverview()
plotStackedOverview(cytoband = TRUE)

## 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

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

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

```
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))
p.new
```

---

scale_fill_fold_change

scale color for fold change values

Description

In biology, lots of data are scaled to value around 0, and people like to show them as blue-white-red scale color, where negative value are blue, 0 is white and positive value is red, and they are scaled for continuous variables.

Usage

```
scale_fill_fold_change()
```

Value

a list.

Author(s)

Tengfei Yin

Examples

```
p1 <- autoplot(volcano - 150)
p1
p1 + scale_fill_fold_change()
```

scale_fill_giemsa *scale filled color to customized giemsa color.*

Description

scale filled color to customized giemsa color.

Usage

```
scale_fill_giemsa(fill = getOption("biovizBase")$cytobandColor)
```

Arguments

fill	a character vector to indicate colors, and names of vector mapped to gieStain name.

Value

a list.

Author(s)

Tengfei Yin

Examples

```
getOption("biovizBase")$cytobandColor
library(biovizBase)
data(hg19IdeogramCyto)
p1 <- autoplot(hg19IdeogramCyto, layout = "karyogram", aes(fill  =
gieStain))
p1
p1 + scale_fill_giemsa()
```

scale_x_sequnit *scale x by unit*

Description

scale x by unit 'Mb','kb', 'bp'.

Usage

```
scale_x_sequnit(unit = c("Mb", "kb", "bp"))
```

Arguments

unit	unit to scale x.

Value

'position_c'

Author(s)

Tengfei Yin

Examples

```
library(ggplot2)
p <- qplot(x = seq(1, to = 10000, length.out = 40), y = rnorm(40), geom
= "point")
## default mb
p + scale_x_sequnit()
p + scale_x_sequnit("kb")
p + scale_x_sequnit("bp")
```

---

stat_aggregate *Generates summaries on the specified windows*

Description

Generates summaries on the specified windows

Usage

```
## S4 method for signature 'GRanges'
stat_aggregate(data, ..., xlab, ylab, main, by, FUN,
                          maxgap=0L, minoverlap=1L,
                          type=c("any", "start", "end", "within", "equal"),
                          select=c("all", "first", "last", "arbitrary"),
                          y = NULL, window = NULL, facets = NULL,
                          method = c("mean", "median","max",
                                   "min", "sum", "count", "identity"),
                          geom = NULL)
```

Arguments

data	A GRanges or data.frame object.
...	Arguments passed to plot function. such as aes() and color.
xlab	Label for x
ylab	Label for y
main	Title for plot.
by	An object with 'start', 'end', and 'width' methods. Passed to aggreagate.
FUN	The function, found via 'match.fun', to be applied to each window of 'x'. Passed to aggreagate.
maxgap, minoverlap	It passed to findOverlaps. Intervals with a separation of maxgap or less and a minimum of minoverlap overlapping positions, allowing for maxgap, are considered to be overlapping. maxgap should be a scalar, non-negative, integer. minoverlap should be a scalar, positive integer.
type	It passed to findOverlaps. By default, any overlap is accepted. By specifying the type parameter, one can select for specific types of overlap. The types correspond to operations in Allen's Interval Algebra (see references). If type is start or end, the intervals are required to have matching starts or ends, respectively. While this operation seems trivial, the naive implementation using outer would be much less efficient. Specifying equal as the type returns the intersection of the start and end matches. If type is within, the query interval must be wholly contained within the subject interval. Note that all matches must additionally satisfy the minoverlap constraint described above. The maxgap parameter has special meaning with the special overlap types. For start, end, and equal, it specifies the maximum difference in the starts, ends or both, respectively. For within, it is the maximum amount by which the query may be wider than the subject.
select	It passed to findOverlaps. When select is "all" (the default), the results are returned as a Hits object. When select is "first", "last", or "arbitrary" the results are returned as an integer vector of length query containing the first, last, or arbitrary overlapping interval in subject, with NA indicating intervals that did not overlap any intervals in subject. If select is "all", a Hits object is returned. For all other select the return value depends on the drop argument. When select != "all" && !drop, an IntegerList is returned, where each element of the result corresponds to a space in query. Whenselect != "all" && drop, an integer vector is returned containing indices that are offset to align with the unlisted query.
y	A character indicate the varialbe column for which aggregation is taken on, same as aes(y =).
window	Integer value indicate window size.
facets	Faceting formula to use.

method	customized method for aggregating, if FUN is not provided.
geom	The geometric object to use display the data.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
library(GenomicRanges)
set.seed(1)
N <- 1000
## ======================================================================
##  simmulated GRanges
## ======================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))


ggplot(gr) + stat_aggregate(aes(y = value))
## or
## ggplot(gr) + stat_aggregate(y = "value")
ggplot(gr) + stat_aggregate(aes(y = value), window = 36)
ggplot(gr) + stat_aggregate(aes(y = value), select = "first")
## Not run:
## no hits
ggplot(gr) + stat_aggregate(aes(y = value), select = "first", type = "within")

## End(Not run)
ggplot(gr) + stat_aggregate(window = 30,  aes(y = value),fill = "gray40", geom = "histogram")
ggplot(gr) + stat_aggregate(window = 100, fill = "gray40", aes(y = value),
                            method = "max", geom = "histogram")

ggplot(gr) + stat_aggregate(aes(y = value), geom = "boxplot")
ggplot(gr) + stat_aggregate(aes(y = value), geom = "boxplot", window = 60)
## now facets need to take place inside stat_* geom_* for an accurate computation
ggplot(gr) + stat_aggregate(aes(y = value), geom = "boxplot", window = 30,
```

```
              facets = sample ~ seqnames)
autoplot(gr, stat = "aggregate", aes(y = value))
autoplot(gr, stat = "aggregate", geom = "boxplot", aes(y = value), window = 36)
```

---

stat_bin *Binning method*

Description

Binning method especially for Rle and RleList, for data.frame it's just calling ggplot2::stat_bin.

Usage

```
## S4 method for signature 'data.frame'
stat_bin(data, ...)

## S4 method for signature 'Rle'
stat_bin(data, ..., binwidth, nbin = 30,
         xlab, ylab, main, geom = c("bar", "heatmap"),
         type = c("viewSums","viewMins",
                                    "viewMaxs", "viewMeans"))
## S4 method for signature 'RleList'
stat_bin(data, ..., binwidth, nbin = 30,
          xlab, ylab, main,
          indName = "sample",
          geom = c("bar", "heatmap"),
          type = c("viewSums","viewMins",
          "viewMaxs", "viewMeans"))
```

Arguments

data	a data.frame or Rle or RleList object.
...	arguments passed to aesthetics mapping.
binwidth	width of the bins.
nbin	number of bins.
xlab	x label.
ylab	y label.
main	title.
indName	when faceted by a RleList, name used for labeling faceted factor. Default is 'sample'.
geom	geometric types.
type	statistical summary method used within bins, shown as bar height or heatmap colors.

Value

a ggplot object.

Author(s)

Tengfei Yin

Examples

```
library(IRanges)
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)

ggplot() + stat_bin(xRle)
ggplot(xRle) + stat_bin()
ggplot(xRle) + stat_bin(nbin = 100)
ggplot(xRle) + stat_bin(binwidth = 200)

p1 <- ggplot(xRle) + stat_bin(type = "viewMeans")
p2 <- ggplot(xRle) + stat_bin(type = "viewSums")
## y scale are different.
tracks(viewMeans = p1, viewSums = p2)

ggplot(xRle) + stat_bin(geom = "heatmap")
ggplot(xRle) + stat_bin(nbin = 100, geom = "heatmap")
ggplot(xRle) + stat_bin(binwidth = 200, geom = "heatmap")

## for RleList
ggplot(xRleList) + stat_bin()
ggplot(xRleList) + stat_bin(nbin = 100)
ggplot(xRleList) + stat_bin(binwidth = 200)

p1 <- ggplot(xRleList) + stat_bin(type = "viewMeans")
p2 <- ggplot(xRleList) + stat_bin(type = "viewSums")
## y scale are different.
tracks(viewMeans = p1, viewSums = p2)

ggplot(xRleList) + stat_bin(geom = "heatmap")
ggplot(xRleList) + stat_bin(nbin = 100, geom = "heatmap")
ggplot(xRleList) + stat_bin(binwidth = 200, geom = "heatmap")
```

---

stat_coverage *Calculate coverage*

Description

Calculate coverage.

Usage

```
# for GRanges
## S4 method for signature 'GRanges'
stat_coverage(data, ..., xlim, xlab, ylab, main,
              facets = NULL, geom = NULL)
# for GRangesList
## S4 method for signature 'GRangesList'
stat_coverage(data, ..., xlim, xlab, ylab, main,
              facets = NULL, geom = NULL)

# for Bamfile
## S4 method for signature 'BamFile'
stat_coverage(data, ..., maxBinSize = 2^14,
               xlim, which, xlab, ylab,
               main, facets = NULL, geom = NULL,
               method = c("estimate", "raw"),
               space.skip = 0.1, coord = c("linear", "genome"))
```

Arguments

data	A GRanges or data.frame object.
...	Extra parameters such as aes() passed to geom_rect, geom_alignment, or geom_segment.
xlim	Limits for x.
xlab	Label for x
ylab	Label for y
main	Title for plot.
facets	Faceting formula to use.
geom	The geometric object to use display the data.
maxBinSize	maxBinSize.
method	'estimate' for parsing estimated coverage(fast), 'raw' is slow and parse the accurate coverage.
which	GRanges which defines region to subset the results.
space.skip	used for coordinate genome, skip between chromosomes.
coord	coordinate system.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
library(ggbio)
## ======================================================================
##  simmulated 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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

ggplot(gr) + stat_coverage()
ggplot() + stat_coverage(gr)

ggplot(gr) + stat_coverage(geom = "point")
ggplot(gr) + stat_coverage(geom = "area")
ggplot(gr) + stat_coverage(aes(y = ..coverage..), geom = "histogram")

ggplot(gr) + stat_coverage(aes(y = ..coverage..)) + geom_point()

## for bam file
## TBD
```

`stat_gene` *Calculate gene structure*

Description

Calculate gene structure.

Usage

```
## S4 method for signature 'TranscriptDb'
stat_gene(data, ...)
```

Arguments

data	A GRanges or data.frame object.
...	Extra parameters such as aes() passed to geom_alignment.

Value

A 'Layer'.

Author(s)

Tengfei Yin

See Also

geom_alignment

Examples

```
## Not run:
## loading package
## Deprecated
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

## made a track comparing full/reduce stat.
p1 <- ggplot() + geom_alignment(txdb, which = genesymbol["RBM17"])
p1 <- ggplot() + stat_gene(txdb, which = genesymbol["RBM17"])
## or
p1 <- ggplot(txdb) + stat_gene(which = genesymbol["RBM17"])

p1 <- ggplot(txdb) + stat_gene(which = genesymbol["RBM17"])
p2 <- ggplot(txdb) + stat_gene(which = genesymbol["RBM17"], stat =
"reduce")
p2 <- ggplot(txdb) + stat_gene(which = genesymbol["RBM17"], stat = "reduce")
## ggplot(txdb) + geom_alignment(which = genesymbol["RBM17"]) + stat_reduce()
## ggplot(txdb) + geom_alignment(which = genesymbol["RBM17"])
tracks(full = p1, reduce = p2, heights = c(3, 1))

## change y labels
ggplot(txdb) + stat_gene(which = genesymbol["RBM17"], names.expr =
"tx_id:::gene_id")

## End(Not run)
```

stat_identity *Transform the data to a data.frame and for multiple geoms.*

Description

Transform the data to a suitable data.frame and then one could use multiple geom or even stat to re-plot the data.

Usage

```
## S4 method for signature 'data.frame'
stat_identity(data, ...)

## S4 method for signature 'GRanges'
stat_identity(data, ..., geom = NULL)

## S4 method for signature 'Rle'
stat_identity(data, ..., xlab, ylab, main, geom = NULL)

## S4 method for signature 'RleList'
stat_identity(data, ..., xlab, ylab, main,
          geom = NULL, indName = "sample")
```

Arguments

data	A GRanges or data.frame object.
...	Extra parameters such as aes() passed to geom_rect, geom_alignment, or geom_segment.
geom	The geometric object to use display the data.
xlab	x label.
ylab	y label.
main	title of graphic..
indName	sample name.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
##  load
set.seed(1)
N <- 50

require(GenomicRanges)
##  simul
## ======================================================================
##  simmulated GRanges
## ======================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

##  geom_point_start
ggplot() + stat_identity(gr, aes(x = start, y = value), geom = "point")
## or more formal
ggplot(gr) + stat_identity(aes(x = start, y = value), geom = "point")

##  geom_point_midpoint
ggplot(gr) + stat_identity(aes(x = midpoint, y = value), geom = "point")

##  geom_rect_all
ggplot(gr) + stat_identity(aes(xmin = start, xmax = end,
                               ymin = value - 0.5, ymax = value + 0.5),
                           geom = "rect")

##  geom_rect_y
ggplot(gr) + stat_identity(aes(y = value), geom = "rect")

##  geom_line
ggplot(gr) + stat_identity(aes(x = start, y = value),  geom = "line")

##  geom_segment
ggplot(gr) + stat_identity(aes(y = value), geom = "segment")

## Rle/RleList
library(IRanges)
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)

ggplot(xRle) + stat_identity(geom = "point")
ggplot(xRleList) + stat_identity(geom = "point")
```

---

stat_mismatch *Calculate mismatch summary*

Description

Calculate mismatch summary

Usage

```
## for GRanges
## S4 method for signature 'GRanges'
stat_mismatch(data, ..., bsgenome,
              xlab, ylab, main,
              geom = c("segment", "bar"),
              show.coverage = TRUE)
## for BamFile
## S4 method for signature 'BamFile'
stat_mismatch(data, ..., bsgenome, which,
              xlab, ylab, main,
              geom = c("segment", "bar"),
              show.coverage = TRUE)
```

Arguments

data	A GRanges or BamFile object.
...	Extra parameters such as aes() passed to geom_rect, geom_alignment, or geom_segment.
bsgenome	BSgenome object.
which	GRanges object to subset the data.
xlab	Label for x
ylab	Label for y
main	Title for plot.
geom	The geometric object to use display the data.
show.coverage	whether to show coverage as background or not.

Value

A 'Layer'.

Author(s)

Tengfei Yin

stat_reduce *Reduce an object.*

Description

Reduce GRanges, IRanges or TranscriptDb object.

Usage

```
## S4 method for signature 'GRanges'
stat_reduce(data, ...,
            xlab, ylab, main,
            drop.empty.ranges = FALSE,
            min.gapwidth = 1L,
            facets = NULL, geom = NULL)

## S4 method for signature 'IRanges'
stat_reduce(data, ...,
            xlab, ylab, main,
            drop.empty.ranges = FALSE,
            min.gapwidth = 1L,
            with.inframe.attrib=FALSE,
            facets = NULL, geom = NULL)

## S4 method for signature 'TranscriptDb'
stat_reduce(data, ...)
```

Arguments

data	GRanges, IRanges or TranscriptDb object.
...	passed to aesthetics mapping.
xlab	x label.
ylab	y label.
main	title.
drop.empty.ranges	pass to reduce function.
min.gapwidth	pass to reduce function.
with.inframe.attrib	pass to reduce function.
facets	pass to reduce function.
geom	geometric type.

Value

a ggplot object.

Author(s)

Tengfei Yin

See Also

reduce.

Examples

```
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

ggplot(gr) + stat_reduce()
autoplot(gr, stat = "reduce")
strand(gr) <- "*"
ggplot(gr) + stat_reduce()

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
## made a track comparing full/reduce stat.
ggplot(txdb) + stat_reduce(which = genesymbol["RBM17"])
```

---

stat_slice *Slice Rle/RleList to view them as bar or heatmap.*

Description

Slice Rle/RleList to different view by set lower or other parameters, then view summary for all those viewed region.

Usage

```
## S4 method for signature 'Rle'
stat_slice(data, ...,
          xlab, ylab, main,
          na.rm = FALSE,
          geom = NULL,
          lower=-Inf, upper=Inf,
          includeLower=TRUE, includeUpper=TRUE,
          rangesOnly = FALSE,
          type = c("viewSums","viewMins",
          "viewMaxs", "viewMeans"))

## S4 method for signature 'RleList'
stat_slice(data, ...,
          xlab, ylab, main,
          indName = "sample",
          na.rm = FALSE,
          geom = NULL,
          lower=-Inf, upper=Inf,
          includeLower=TRUE, includeUpper=TRUE,
          rangesOnly = FALSE,
          type = c("viewSums","viewMins",
                   "viewMaxs", "viewMeans"))
```

Arguments

data	a data.frame or Rle or RleList object.
...	arguments passed to aesthetics mapping.
xlab	x label.
ylab	y label.
main	title.
indName	when faceted by a RleList, name used for labeling faceted factor. Default is 'sample'.
geom	geometric types.
type	statistical summary method used within bins, shown as bar height or heatmap colors.
na.rm	logical value, default FALSE, passed to function like viewMaxs for statistical summary computation.
lower	passed to slice.
upper	passed to slice.
includeLower	passed to slice.
includeUpper	passed to slice.
rangesOnly	passed to slice.

Value

a ggplot object.

Author(s)

Tengfei Yin

See Also

slice

Examples

```
library(IRanges)
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)

ggplot(xRle) + stat_slice(lower = 5)
ggplot(xRle) + stat_slice(lower = 5, geom = "bar")
ggplot(xRle) + stat_slice(lower = 5, geom = "heatmap")

p1 <- ggplot(xRle) + stat_slice(type = "viewMeans", lower = 5,
                                geom = "bar")
p2 <- ggplot(xRle) + stat_slice(type = "viewSums", lower = 5,
                                geom = "bar")
## y scale are different.
tracks(viewMeans = p1, viewSums = p2)

ggplot(xRleList) + stat_slice(lower = 5)
ggplot(xRleList) + stat_slice(lower = 5, geom = "bar")
ggplot(xRleList) + stat_slice(lower = 5, geom = "heatmap")

p1 <- ggplot(xRleList) + stat_slice(type = "viewMeans", lower = 5,
                                geom = "bar")
p2 <- ggplot(xRleList) + stat_slice(type = "viewSums", lower = 5,
                                geom = "bar")
## y scale are different.
tracks(viewMeans = p1, viewSums = p2)
```

---

stat_stepping *Calculate stepping levels*

Description

Calculate stepping levels.

Usage

```
## S4 method for signature 'GRanges'
stat_stepping(data, ..., xlab, ylab, main,
              facets = NULL,
              geom = c("rect", "alignment", "segment"))
```

Arguments

data	A GRanges or data.frame object.
...	Extra parameters such as aes() passed to geom_rect, geom_alignment, or geom_segment.
xlab	Label for x
ylab	Label for y
main	Title for plot.
facets	Faceting formula to use.
geom	The geometric object used to display the data. For 'stepping', could be one of 'rect', 'alignment', 'segment'.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
set.seed(1)
N <- 50

require(GenomicRanges)
##  simul
## ========================================================================
##  simmulated GRanges
## ========================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))
```

```
##  default
ggplot(gr) + stat_stepping()
## or
ggplot() + stat_stepping(gr)

##  facet_aes
ggplot(gr) + stat_stepping(aes(color = strand, fill = strand),
                                facets = sample ~ seqnames)

##  geom_segment
ggplot(gr) + stat_stepping(aes(color = strand),
                    geom = "segment", xlab = "Genomic coord", ylab = "y", main = "hello")


##  geom_alignment
## ggplot(gr) + stat_stepping(geom = "alignment")

##  geom_alignment_group
##  ggplot(gr) + stat_stepping(aes(group = pair),geom = "alignment")
```

`stat_table` *Tabulate a GRanges object*

Description

Tabulate a GRanges object

Usage

```
## S4 method for signature 'GRanges'
stat_table(data, ..., xlab, ylab, main,
            geom = NULL,stat = NULL)
## S4 method for signature 'GRangesList'
stat_table(data, ..., xlab, ylab, main,
            facets = NULL, geom = NULL)
```

Arguments

`data`	A `GRanges` or `data.frame` object.
`...`	Extra parameters such as aes() passed to `geom_rect`, `geom_alignment`, or `geom_segment`.
`xlab`	Label for x
`ylab`	Label for y
`main`	Title for plot.
`facets`	Faceting formula to use.
`geom`	The geometric object to use display the data.
`stat`	The geometric object to use display the data.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
##  load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
##  simul
## ======================================================================
##  simmulated GRanges
## ======================================================================
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),
              sample = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))

gr <- c(gr[seqnames(gr) == "chr1"][sample(1:10, size = 1e4, replace = TRUE)],gr)

##  default
ggplot(gr) + stat_table()
ggplot(gr) + stat_table(geom = "segment", aes(y = ..score.., color = ..score..))
ggplot(gr) + stat_table(aes(color = score))
```

---

theme *theme in ggbio*

Description

Theme defined in ggbio for plot or tracks.

Usage

```
theme_null()
theme_noexpand()
theme_alignment(ylabel = FALSE, base_size = 12, base_family = "",
axis = TRUE, border = TRUE, grid = TRUE)
theme_pack_panels(strip.bg = FALSE, strip.text.y = TRUE)
theme_clear(grid.y = FALSE, grid.x.minor = FALSE, grid.x.major = FALSE,
            panel.background.fill = "white", panel.border.color = NA,
            axis.ticks.x = FALSE, axis.ticks.y = TRUE, grid.color = "gray95",
            axis.line.color = "gray80")
theme_tracks_sunset(bg = "#fffedb", alpha = 1, ...)
theme_genome()
```

Arguments

alpha	alpha blending from 0(transparent) to 1(solid).
axis	logical value, show axis or not.
axis.line.color	color for axis line .
axis.ticks.x	show x ticks or not.
axis.ticks.y	show y ticks or not.
base_family	family for font.
base_size	size for font.
bg	background color for tracks.
border	logical value, show border or not.
grid	logical value, show background grid or not.
grid.color	grid line color.
grid.x.major	show x major grid line or not.
grid.x.minor	show x minor grid line or not.
grid.y	show y grid or not.
panel.background.fill	panel background fill color.
panel.border.color	panel border color.
strip.bg	if strip background is removed.
strip.text.y	if strip text is removed.
ylabel	logical value. Show labels or not.
...	passed to theme_clear.

Details

Themes speciall designed for tracks, are named following naming schema theme_tracks_*

Value

Return a theme.

Author(s)

Tengfei Yin

Examples

```
##  load
library(ggbio)
p <- qplot(data = mtcars, x = mpg,  y = wt, facets = cyl ~ .)
p + theme_null()
p + theme_clear()
p + theme_pack_panels()
p + theme_alignment()
p1 <- qplot(data = mtcars, x = mpg,  y = wt)
tracks(p1 = p, p2 = p1)
tracks(p1 = p, p2 = p1) + theme_tracks_sunset()
```

---

tracks *Tracks for genomic graphics*

Description

tracks is a conventient constructor for bindind graphics as trakcs. You dont' have to worry about adjusting different graphics, tracks did that for you. It's NOT just limited to bind genomic tracks, you can use this function to bind any tracks with the same defination of x axis, for example, sets of time series plots you made.

Tracks view is most common way to viewing genome features and annotation data and widely used by most genome browsers. Our assumption is that, most graphics you made with ggbio or by yourself using ggplot2, are almost always sitting on the genomic coordinates or the same x axis. And to compare annotation information along with genome features, we need to align those plots on exactly the same x axis in order to form your hypothesis. This function leaves you the flexibility to construct each tracks separately with worrying your alignments later.

Usage

```
tracks(..., heights, xlim, xlab = NULL, main = NULL,
            title = NULL, theme = NULL, fixed = NULL,
            track.plot.color = NULL,
            track.bg.color = NULL,
            main.height = unit(2, "lines"),
            scale.height = unit(2, "lines"),
            xlab.height = unit(2, "lines"),
            padding = unit(-1, "lines"),
            label.bg.color =  "white",
```

```
            label.bg.fill = "gray80",
            label.text.color = "black",
            label.text.cex = 1,
            label.width = unit(2.5, "lines"))
```

Arguments

...	plots of class ggplot, generated from ggplot2 or ggbio.
heights	numeric vector of the same length of passed graphic object to indicate the ratio of each track.
xlim	limits on x. could be IRanges, GRanges, numeric value
xlab	label for x axis.
main	title for the tracks.
title	title for the tracks, alias like main.
theme	theme object used for building tracks, this will set to default, which could be reseted later.
fixed	vector of logical value, indicates whether the scale of passed graphics are fixed or not, useful for attaching unchanged plots to the tracks, like ideogram. Please check utilities section below.
track.plot.color	Vector of characters of length 1 or the same length of passed plots, background color for each track, default is white.
track.bg.color	background color for the whole tracks.
main.height	unit. Height to control the title track height.
scale.height	unit. Height to control the scale track height.
xlab.height	unit. Height to control the xlab track height.
padding	single numeric value or unit, if numeric value, the unit would be "lines" by default.
label.bg.color	track labeling background rectangle border color.
label.bg.fill	track labeling background fill color.
label.text.color	track labeling text color.
label.text.cex	track labeling text size.
label.width	track labeling size.

Details

tracks did following modification for passed plots.

- remove x-axis, ticks, xlab and tile for each track and add scales at bottom. We suppose a new xlab and title would be provided by the tracks function for the whole tracks, but we still keep individual's y axis.
- align x-scale limits to make sure every plots sitting on exactly the same x scale.
- squeezing plots together to some extent.
- labeling tracks if names are provided, please check utilities section about labeled method.
- return a track object. This would allow many features introduced in this manual.

Value

A Tracks object.

Track class

constructor tracks will return a Tracks object, which has following slots.

grobs a ggplotGrobList object contains a list of ggplot object, which is our passed graphics.

backup a backup of all the slots for holding the original tracks, so users could edit it and reset it back at any time later, and backup method will reset the backupped copy.

ylim y limits for each plot.

labeled vector of logical value indicates whether a track is labeled or not, for labeled attributes please check utilities section.

mutable vector of logical value indicates whether a track is mutable for theme editing or not, for mutable attributes please check utilities section.

hasAxis vector of logical value indicates whether a track has axis or not, for hasAxis attributes please check utilities section.

heights, xlim, xlab, main, title, theme, fixed, track.plot.color, track.bg.color, main.height, scal
those slots are described in arguments section for constructor.

Utilities

Please check examples for usage.

summary(object) summary information about tracks object.

fixed(x), fixed(x) <- value x is the ggplot object, this controls if a track has a fixed x scale or not, if the fixed attributes is TRUE, then when you pass this plot to a tracks, this plot won't be re-aligned with other tracks and will keep the original x-axis, this allow you to pass some plot like ideogram. fixed function will return a logical value

labeled(x), labeled(x) <- value x is the ggplot object, if you pass named graphics into tracks, it will create the labels on the left for you. Several ways supported to name it. You can pass a list of graphics with names. Or you can use tracks('name1' = p1, 'name 2' = p2, ...) with quotes for complicated words or simply tracks(part1 = p1, part = p2, ...).

mutable(x), mutable(x) <- value x is the ggplot object, this controls whether a plot in the tracks mutable to theme changing or not, when you use + method for Tracks object, add-on edit will only be applied to the the mutable plots.

bgColor(x), bgColor(x) <- value x is the ggplot object, this change the background color for single plot shown in the tracks.

xlim(x), xlim(x) <- value when x is the numeric value, it calls ggplot2::coord_cartesian(xlim = ...) method, we doesn't use ggplot2::xlim() for the reason it will cut data outside the range, and we believe the best behavior would be zoom-in/out like most browser. when x is IRanges, GRanges, it get the range and passed to ggplot2::coord_cartesian function.

when x is Tracks object, xlim(x) will return x limits for that tracks. xlim(x) <- value replace method only works for Tracks object. value could be numeric, IRanges, GRanges object. This will change the x limits associated with tracks.

+ `xlim(obj)`:`obj` is the numeric range, or `IRanges`, `GRanges` object.

+ `coord_cartesian()`: please read manual in ggplot2, this controls both xlim an ylim, only accept numerical range.

+ The most nice features about `Tracks` object is the one inherited from ggplot2's components additive features, with + method you can use any `theme` object and utilities in ggplot2 package, to add them on a `Tracks` object, for example, if x is our `Tracks` object, `x + theme` would apply theme to any plots in the tracks except those are immutable.

Backup and reset

reset(obj) `obj` is the Tracks object, this reset the tracks back to original or backuped version.

backup(obj) `obj` is the Tracks object, this clear previous backup and use current setting for a new backup.

update(obj, xlim) `obj` is the Tracks object, this allow a quick tweak with xlim when the plot is shown on your screen.

Author(s)

Tengfei Yin

See Also

`align.plots`

Examples

```
## make a simulated time series data set
df1 <- data.frame(time = 1:100, score = sin((1:100)/20)*10)
p1 <- qplot(data = df1, x = time, y = score, geom = "line")
df2 <- data.frame(time = 30:120, score = sin((30:120)/20)*10, value = rnorm(120-30 + 1))
p2 <- ggplot(data = df2, aes(x = time, y = score)) +
  geom_line() + geom_point(size = 4, aes(color = value))
## check p2
p1
## check p2
p2

## binding
tracks(p1, p2)

## or
tks <- tracks(p1, p2)
tks

## labeling: default labeling a named graphic
## simply pass a name with it
tracks(time1 = p1, time2 = p2)
## or pass a named list with it
lst <- list(time1 = p1, time2 = p2)
tracks(lst)
```

```
## more complicated case please use quotes
tracks(time1 = p1, "second time" = p2)

## set heights
tracks(time1 = p1, time2 = p2, heights = c(1, 3))

## if you want to disable label arbitrarily
## default label is always TRUE
labeled(p2)
labeled(p2) <- FALSE
## set labeled to FALSE, remove label even the plot has a name
tracks(time1 = p1, time2 = p2)
labeled(p2) <- TRUE

## fix a plot, not synchronize with other plots
p3 <- p1
## default is always FALSE
fixed(p3)
## set to TRUE
fixed(p3) <- TRUE
fixed(p3)

tracks(time1 = p1, time2 = p2, "time3(fixed)" = p3)


fixed(p3) <- FALSE
## otherwise you could run
tracks(time1 = p1, time2 = p2, "time3(fixed)" = p3, fixed = c(FALSE, FALSE, TRUE))


## control axis
hasAxis(p1)
hasAxis(p1) <- TRUE
# ready for weird looking
tracks(time1 = p1, time2 = p2)
# set it back
hasAxis(p1) <- FALSE



## mutable
mutable(p1)
tracks(time1 = p1, time2 = p2) + theme_bw()
mutable(p1) <- FALSE
# mutable for "+" method
tracks(time1 = p1, time2 = p2) + theme_bw()
mutable(p1) <- TRUE

## bgColor
bgColor(p1)
tracks(time1 = p1, time2 = p2)
bgColor(p1) <- "brown"
```

```
# mutable for "+" method
tracks(time1 = p1, time2 = p2)
# set it back
bgColor(p1) <- "white"

## apply a theme to each track
tks <- tracks(time1 = p1, time2 = p2) + theme_bw()
tks
reset(tks)

## store it with tracks
tks <- tracks(time1 = p1, time2 = p2, theme = theme_bw())
tks
tks <- tks + theme_gray()
tks
## reset will be introduced later
reset(tks)

## apply a pre-defiend theme for tracks!
tracks(time1 = p1, time2 = p2) + theme_tracks_sunset()
 tracks(p1, p2) + theme_tracks_sunset()

## change limits
tracks(time1 = p1, time2 = p2) + xlim(c(1, 40))
tracks(time1 = p1, time2 = p2) + xlim(1, 40)
tracks(time1 = p1, time2 = p2) + coord_cartesian(xlim = c(1, 40))
# change y
tracks(time1 = p1, time2 = p2) + xlim(1, 40) + ylim(0, 10)
library(GenomicRanges)
gr <- GRanges("chr", IRanges(1, 40))
# GRanges
tracks(time1 = p1, time2 = p2) + xlim(gr)
# IRanges
tracks(time1 = p1, time2 = p2) + xlim(ranges(gr))
tks <- tracks(time1 = p1, time2 = p2)
xlim(tks)
xlim(tks) <- c(1, 35)
xlim(tks) <- gr
xlim(tks) <- ranges(gr)

## xlab, title
tracks(time1 = p1, time2 = p2, xlab = "time")
tracks(time1 = p1, time2 = p2, main = "title")
tracks(time1 = p1, time2 = p2, title = "title")
tracks(time1 = p1, time2 = p2, xlab = "time", title = "title") + theme_tracks_sunset()


## backup and restore
tks <- tracks(time1 = p1, time2 = p2)
tks
tks <- tks + xlim(1, 40)
tks
reset(tks)
```

```
tks <- tks + xlim(1, 40)
tks
tks <- backup(tks)
tks <- tks + theme_bw()
tks
reset(tks)

## update
## show it on the fly, save your time to re-print
tks <- tracks(time1 = p1, time2 = p2)
tks
update(tks, xlim = c(1, 40))

## padding(need to be fixed for more delicate control)
tracks(time1 = p1, time2 = p2, padding = 2)

## track color
tracks(time1 = p1, time2 = p2, track.bg.color = "yellow")
tracks(time1 = p1, time2 = p2, track.plot.color = c("yellow", "brown"))

## alignPlots simply align the panel, without adjusting the xlims
alignPlots(p1, p2)

alignPlots(p1, p2 + theme(legend.position = "none"), vertical = FALSE)
```

Index