Package 'ggbio'

September 4, 2012

Version 1.4.6

Title Static visualization 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)

Imports methods, biovizBase(>= 1.4.2), reshape2, ggplot2(>= 0.9),BiocGenerics, Biobase, IRanges, GenomicRanges, GenomicFeatures,Rsamtools, BSgenome, gridExtra, scales, plyr,VariantAnnotation, Hmisc, rtracklayer

Suggests BSgenome.Hsapiens.UCSC.hg19,TxDb.Hsapiens.UCSC.hg19.knownGene, affyPLM

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 theme.R tracks.R fortify-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 autoplot-method.R plotGrandLinear.R plotStackedOverview.R plotRangesLinkedToData.R plotFragLength-method.R plotSpliceSum-method.R rescale-method.R

R topics documented:

`align.plots` *align plots on x axis*

Description

align plots on x axis

Usage

```
align.plots(..., vertical = TRUE,
            heights = unit(rep(1, nrow), "null"))
```

Arguments

`...`	ggplot2 graphic object.
`vertical`	logical value indicate how to align the plots, currently only support vertical alignments.
`heights`	numeric values or units, indicate the heights of each track.

Value

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

Author(s)

Tengfei Yin, Baptiste Auguie

See Also

tracks

Examples

```
## @knitr load
## ============================================================
## Load packages
## ============================================================
## Load gene features for human
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

## @knitr tracks
## ============================================================
## Create tracks
## ============================================================
## create two tracks
## full gene model
p1 <- ggplot() + stat_gene(txdb, which = genesymbol["RBM17"], geom = "gene")
## reduced gene model
p2 <- ggplot() + stat_gene(txdb, which = genesymbol["RBM17"], geom = "reduced_gene")
## @knitr align.plots
## ============================================================
## align.plots
## ============================================================
```

---

autoplot *Generic autoplot function*

Description

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

Usage

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

```
                c("linear", "karyogram", "circle"))


## For object GRangesList
## 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)


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


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

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

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

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


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

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


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


## For object ExpressionSet
## S4 method for signature 'ExpressionSet'
autoplot(object, ..., type = c("none", "heatmap",
                                    "matrix", "parallel", "MA",
                                    "mean-sd", "volcano",
                                    "NUSE", "RLE"),
                     test.method = "t")

## For object GenomicRangesList
## S4 method for signature 'GenomicRangesList'
autoplot(object, args = list(), trackWidth,
                    radius = 10, grid = FALSE, trackSkip = 1, layout = c("circle"))
## For object VCF
## S4 method for signature 'VCF'
autoplot(object, ..., xlab, ylab, main,
                    type = c("geno", "info", "fixed"),
                    ylabel = TRUE)
```

Arguments

object	object to be plot.
x	x value, start/end/midpoint without quotes. e.g `x = start`, default use the `midpoint`.
y	y value, only be used in geom: point/line. Should be a single name of the column names in the elementMetatata without quotes. e.g `y = score`
geom	Geom to use (Single character for now). Please see section Geometry for details.
size	Size for point or lines. Could equal a column name or a fixed number. When it's fixed, please use `I()` to wrap the value.
shape	Shape for point or lines. Could equal a column name or a fixed number. When it's fixed, please use `I()` to wrap the value.
color	Color for point for lines. Could equal a column name or a fixed character. When it's fixed, please use `I()` to wrap the value.

alpha	Alpha blending. Could equal a column name or a fixed number. When it's fixed, please use I() to wrap the value.
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.
lower	When object is Rle/RleList, and use other types of methods which is not "raw", at least a lower parameters which passed to slice function is required.
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.
args	a list of arguments list which applied to each track.
trackWidth	Numeric values indicate width for each track, if it's only one value, then applied the same value to each track, otherwise, the track must be of the same length of track numbers.
radius	numeric value indicate inner radius of the innermost circle.
grid	logical value of length 1 or the same lengths of tracks numbers, indicate whether you want to adding grid background or not.
trackSkip	numeric values indicate skipped region between tracks.
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.
ylabel	logical value indicates to show y labels or not.
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.

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 <- 200
library(GenomicRanges)
gr <- GRanges(seqnames = sample(c("chr1", "chr2", "chr3"),
                      size = N, replace = TRUE),
              IRanges(start = sample(1:300, size = N, replace = TRUE),
                      width = sample(70:75, size = N,replace = TRUE)),
              strand = sample(c("+", "-", "*"), size = N,
                replace = TRUE),
              value = rnorm(N, 10, 3), score = rnorm(N, 100, 30),
              group = sample(c("Normal", "Tumor"),
                size = N, replace = TRUE),
              pair = sample(letters, size = N,
                replace = TRUE))
autoplot(gr)
autoplot(gr, geom = "segment")
autoplot(gr, geom = "arrowrect")
autoplot(gr, geom = "line", aes(y = value))
autoplot(gr, geom = "point", aes(y = value))

autoplot(gr, facets = group ~ seqnames)

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
autoplot(gr.b, geom = "bar", aes(fill = value))
autoplot(gr.b, geom = "bar", aes(y = value))
ggplot() + layout_circle(gr.b, geom = "bar")

## character
## @knitr bed
library(rtracklayer)
test_path <- system.file("tests", package = "rtracklayer")
test_bed <- file.path(test_path, "test.bed")
wh <- GRanges("chr7", IRanges(127472000, 127473000))
autoplot(test_bed)
autoplot(test_bed, aes(fill = score), geom  = "rect")
autoplot(test_bed, aes(fill = name))
autoplot(test_bed, aes(fill = name), which = wh)

## more object supported, please check the on-line documentation.
```

`fortify` *Fortify a object to a data frame*

Description

Fortify a object to a data frame.

Usage

```
## S4 method for signature 'eSet,missing'
fortify(model, data)
## S4 method for signature 'GRanges,missing'
fortify(model, data)
```

Arguments

`model`	model or other R object to convert to data frame
`data`	Origainal data if needed.

Value

a data.frame object.

Author(s)

Tengfei Yin

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"),
               main.geom = c("rect", "arrowrect"),
               gap.geom = c("chevron", "arrow", "segment"),
               rect.height = NULL,  group.selfish = TRUE)
```

Arguments

data	A GRanges or data.frame 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.
main.geom	Geom for 'main' intervals which is the data you passed.
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.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
## @knitr load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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))


## @knitr default
## ======================================================================
##  default
## ======================================================================
ggplot() + geom_alignment(gr)

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

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

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

## @knitr main_gap
## ======================================
##  main/gap geom
## ======================================
ggplot() + geom_alignment(gr, main.geom = "arrowrect", gap.geom = "chevron")
```

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

```
## @knitr load
set.seed(1)
N <- 100
library(ggbio)
library(GenomicRanges)

## @knitr 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))

## @knitr default
## ======================================
##  default
## ======================================
ggplot() + geom_arch(gr)

## @knitr facet_aes
## ======================================
##  facetting and aesthetics
## ======================================
ggplot() + geom_arch(gr, 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 = 0.05,
              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

```
## @knitr load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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))

## @knitr default
## ============================================================================
##  default
## ============================================================================
ggplot() + geom_arrow(gr)

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

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

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

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


## @knitr options
## ============================================================================
## other options to control arrow angle, density, ...
## ============================================================================
library(grid)
ggplot() + geom_arrow(gr, 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,
                           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.
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

```
## @knitr load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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))


    ## @knitr default
    ## ======================================================================
    ##  default
    ## ======================================================================
    ggplot() + geom_arrowrect(gr)

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

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

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

    ## @knitr group.selfish
    ## ======================================================================
    ##  group.selfish controls when
    ## ======================================================================
    ggplot() + 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

```
## @knitr load
library(ggbio)
library(GenomicRanges)

## @knitr 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

## @knitr bar
ggplot() + geom_bar(gr.b, aes(fill = value))
ggplot() + geom_bar(gr.b,  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

```
## @knitr load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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))


## @knitr default
## ======================================================================
##  default
## ======================================================================
ggplot() + geom_chevron(gr)

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

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

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

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


## @knitr offset
## ========================================================================
##  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)
## @knitr offset:default
ggplot() + geom_rect(gr2) + geom_chevron(gr2.p)

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

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

## @knitr chevron.height:default
## ========================================================================
##  chevron.height
## ========================================================================
values(gr2.p)$score <- c(100, 200)
ggplot() + geom_rect(gr2) + geom_chevron(gr2.p, offset = "score")
## chevron.height
ggplot() + geom_rect(gr2) + 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

```
## @knitr load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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))
```

```
## @knitr data.frame
## ========================================================================
##  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))


## @knitr default
## ========================================================================
##  default
## ========================================================================
ggplot() + geom_rect(gr)

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

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

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

## @knitr group.selfish
## ========================================================================
##  group.selfish controls when
## ========================================================================
ggplot() + geom_rect(gr, 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

```
## @knitr load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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))


## @knitr data.frame
```

```
## ========================================================================
##  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))


## @knitr default
## ========================================================================
##  default
## ========================================================================
ggplot() + geom_segment(gr)

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

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

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

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

`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", "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,
                scale.n = 60, scale.unit = NULL, scale.type = c("M",
```

```
"B", "sci"), grid.n = 5, grid.background = "gray70",
grid.line = "white", grid = FALSE)
```

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.
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.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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))

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

`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 = NULL, 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.
`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

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

## @knitr 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)

## @knitr coord:genome
autoplot(gr, coord = "genome")
autoplot(gr, coord = "genome", aes(fill = seqnames))
gr.t <- transformToGenome(gr)
head(gr.t)

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


## @knitr 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)
                  )

## @knitr 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)



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

## @knitr 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)

## @knitr 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)

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



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

## @knitr 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")

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

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

## @knitr 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)

## @knitr rotate
plotGrandLinear(gr.snp, aes(y = pvalue)) + opts(axis.text.x=theme_text(angle=-90, hjust=0))
```

---

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 = "Genomic Coordinates",
                ylab = substitute(y), main, theme)
```

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.
theme	theme options.

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

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

## @knitr 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)

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

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


## @knitr 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)
                  )

## @knitr 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)



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

## @knitr 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)

## @knitr 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)

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



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

## @knitr 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")

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

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

## @knitr 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)

## @knitr rotate
plotGrandLinear(gr.snp, aes(y = pvalue)) + opts(axis.text.x=theme_text(angle=-90, hjust=0))
```

```
## @knitr 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,
                                    track.skip = -1,
                                    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.
track.skip	skip between tracks.
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

```
## Not run:
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(T,F), 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("red",
"gray90"))

## End(Not run)
```

plotSingleChrom *Plot single chromosome with cytoband*

Description

Plot single chromosome with cytoband

Usage

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

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.

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
plotSingleChrom(hg19IdeogramCyto, subchr = "chr1")
plotSingleChrom(hg19IdeogramCyto, subchr = "chr1", xlabel = TRUE)
## zoom
plotSingleChrom(hg19IdeogramCyto, subchr = "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))
```

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

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

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

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.
wise	wise: If 'TRUE' will wisely expand the actual range of the plot a little, in the way that setting the limits on the scales does

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
```

---

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, start = NULL,
                 end = NULL, width = NULL, y = NULL, frequency = NULL,
                 delta = NULL, simplify = TRUE, window = NULL, facets =
                 NULL, type = c("mean", "median", "max", "min", "sum",
                 "count", "identity"), geom = NULL)
```

Arguments

data	A GRanges or data.frame object.
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.
start	Start of the window. If 'by' is missing, then must supply two of the 'start', 'end', 'width'. If 'window' is provided then you don't have to specify it.
end	End of the window. If 'by' is missing, then must supply two of the 'start', 'end', 'width'. If 'window' is provided then you don't have to specify it.
width	Width of the window. If 'by' is missing, then must supply two of the 'start', 'end', 'width'. If 'window' is provided then you don't have to specify it.
y	A character indicate the varialbe column for which aggregation is taken on. Notice for geom like 'boxplot', we don't compute or aggregate the variable, we simply want to use the idenitcal y as y axis, in that case please put y in the aes mapping function.
frequency	Optional arguments that specify the sampling frequency within the window.
delta	Optional arguments that specify the sampling increment within the window.
...	Arguments passed to plot function. such as aes() and color.
simplify	A logical value specifying whether or not the result should be simplified to a vector or matrix if possible.
window	Integer value indicate window size.
facets	Faceting formula to use.
type	
geom	The geometric object to use display the data.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
## @knitr load
library(ggbio)
library(GenomicRanges)

## @knitr simul
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))

## 36,37 doesn't work
ggplot() + stat_aggregate(gr, y = "value",fill = "gray40")
ggplot() + stat_aggregate(gr, window = 30,  y = "value",fill = "gray40", geom = "histogram")
ggplot() + stat_aggregate(gr, window = 100, fill = "gray40",y = "value",
                          type = "max", geom = "histogram")
```

`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"))
```

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.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
## @knitr load
library(ggbio)

## @knitr simul
## ======================================================================
##  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))


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

---

stat_gene *Calculate gene structure*

Description

Calculate gene structure.

Usage

```
## S4 method for signature 'TranscriptDb'
stat_gene(data, ..., which, xlim, truncate.gaps = FALSE,
                truncate.fun = NULL, ratio = 0.0025, xlab, ylab, main,
                facets = NULL, geom = "gene", stat = c("identity",
                "reduce"), names.expr = expression(paste(tx_name, "(",
                gene_id, ")", sep = "")))
```

Arguments

data	A GRanges or data.frame object.
...	Extra parameters such as aes() passed to geom_rect, geom_alignment, or geom_segment.
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.
xlab	Label for x
ylab	Label for y
main	Title for plot.
facets	Faceting formula to use.
geom	geometric object. only support "gene" now.
stat	defualt "identity" give full gene model and "reduce" for reduced model.
names.expr	Expression for showing y label.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
## @knitr load
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

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

---

stat_identity *Calculate coverage*

Description

Calculate coverage.

Usage

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

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

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.

Value

A 'Layer'.

Author(s)

Tengfei Yin

Examples

```
## @knitr load
set.seed(1)
N <- 50
require(ggbio)
require(GenomicRanges)
## @knitr 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))

## @knitr geom_point_start
ggplot() + stat_identity(gr, aes(x = start, y = value), geom = "point")

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

## @knitr geom_rect_all
ggplot() + stat_identity(gr, aes(xmin = start, xmax = end,
```

```
                                ymin = value - 0.5, ymax = value + 0.5),
                            geom = "rect")

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

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

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

`stat_mismatch` *Calculate mismatch summary*

Description

Calculate mismatch summary

Usage

```
## for GRanges
## S4 method for signature 'GRanges'
stat_mismatch(data, ..., bsgenome, which,
              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_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

```
## @knitr load
set.seed(1)
N <- 50
require(ggbio)
require(GenomicRanges)
## @knitr 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))

## @knitr default
ggplot() + stat_stepping(gr)

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

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


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

## @knitr geom_alignment_group
ggplot() + stat_stepping(gr, 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

```
## @knitr load
set.seed(1)
N <- 100
require(ggbio)
require(GenomicRanges)
## @knitr 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)

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

---

theme_alignment *Theme for alignment*

Description

Theme for alignment

Usage

```
theme_alignment(label = FALSE, base_size = 12,
                base_family = "", axis = TRUE,
                border = TRUE, grid = TRUE)
```

Arguments

label	logical value. Show labels or not.
base_size	size for font
base_family	family for font
axis	logical value, show axis or not.
border	logical value, show border or not.
grid	logical value, show background grid or not.

Value

Return a options list.

Author(s)

Tengfei Yin

Examples

```
## @knitr load
library(ggbio)
data(genesymbol, package = "biovizBase")
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

## @knitr theme:default
p <- autoplot(txdb, which = genesymbol["RBM17"])
p

## @knitr theme:tweak
p + theme_alignment(border = TRUE, grid = FALSE, label = TRUE)
```

---

theme_null *Blank theme*

Description

Theme with no background, axis, legend. Useful for circular plot.

Usage

```
theme_null()
```

Value

Return a options list.

Author(s)

Tengfei Yin

Examples

```
## @knitr load
set.seed(1)
N <- 50
require(ggbio)
require(GenomicRanges)
## @knitr simul
## ======================================================================
##  simmulated GRanges
## ======================================================================
gr <- GRanges(seqnames = "chr1",
              IRanges(start = sample(1:300, size = N, replace = TRUE),
                      width = sample(70:75, size = N,replace = TRUE)),
              strand = sample(c("+", "-", "*"), size = N,
                replace = TRUE))

## @knitr default
autoplot(gr)

## @knitr theme_null
autoplot(gr) + theme_null()
```

---

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.

ggbio provide a set of utilities to reset, backup, and apply options to tracks, please see examples below.

Usage

```
tracks(..., heights, xlim, xlab = NULL,
           opts = NULL, track.skip = -1,
           xlim.change = rep(TRUE, length(list(...))),
           track.plot.col = rep("white", nrow))

## S4 method for signature 'Tracks'
```

```
summary(object)
## S4 method for signature 'Tracks'
print(x)
## S4 method for signature 'Tracks'
show(object)
## S4 method for signature 'Tracks,ANY'
Arith(e1, e2)
## S4 method for signature 'numeric'
xlim(obj, ...)
## S4 method for signature 'Tracks'
xlim(obj, ...)
xlim(x) <- value
## S4 method for signature 'Tracks'
update(object, xlim)
## S4 method for signature 'Tracks'
reset(obj)
## S4 method for signature 'Tracks'
backup(obj)
```

Arguments

...	plots of class ggplot2, trellis, or grobs, and valid arguments to grid.layout.
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.
opts	Option list or theme applied to each track.
track.skip	Numeric value, skip between tracks, unit is 'lines'.
xlim.change	Vector of logical value of the same length as passed graphic objects, to control whether we adjust that track with each other or just leave it as it is. This could be useful when you pass a single chromosome view on top of the tracks.
track.plot.col	plot background color for each track, default is white
obj	Tracks object.
object	Tracks object.
x	Tracks object.
value	Replaced xlim value.
e1	Tracks object on the left of '+'.
e2	option object like in 'ggplot2', on the right of '+'.

Details

tracks function has some extra special features.

- Only keep the bottom x axis based on assumption that all tracks are on the same space, but still keep x ticks. For simply wrapping, please use 'align.plots'.
- 'ncol' which defines columns is always 1, because binding tracks in the context of genomic data is almost always one single column. Multiple column alignments are not supported yet.

also has some utilities.

- xlim
- reset,backup `reset` and backup help you play with options and appearance of the tracks, you could save certain status by calling backup, and get backup version back by calling reset.
- summary `summary` give you meta information about tracks.
- update `update` allow you to update a plot xlim on the fly, you can simply keep the plot window and run `update` to tweak with the view. Other wise you need to revise the tracks object and print it again.
- show,print show plots on your screen.

Value

A `Tracks` object.

Author(s)

Tengfei Yin

See Also

`align.plots`

Examples

```
## @knitr load
## ============================================================
## Load packages
## ============================================================
## Load gene features for human
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
data(genesymbol, package = "biovizBase")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

## @knitr tracks
## ============================================================
## Create tracks
## ============================================================
## create two tracks
## full gene model
p1 <- ggplot() + stat_gene(txdb, which = genesymbol["RBM17"], geom = "gene")
## reduced gene model
p2 <- ggplot() + stat_gene(txdb, which = genesymbol["RBM17"], geom = "reduced_gene")
## building tracks
obj <- tracks(p1, p2, heights = c(3, 1))
## showing
obj


## @knitr align.plots
## ============================================================
## align.plots
## ============================================================
align.plots(p1, p2)
```

```
## @knitr reset
## ============================================================
## test reset/backup
## ============================================================
## create tracks
obj <- tracks(p1, p2, heights = c(3, 1))
## show it
obj
## three ways to change x limits, IRanges/GRanges/numeric
xlim(obj) <- IRanges(start = 6145000, end = 6150000)
xlim(obj) <- GRanges("chr1", c(start = 6145000, end = 6150000))
xlim(obj) <- c(6145000, 6150000)
## show it
obj
## reset to original setting
obj <- reset(obj)
## get back
obj
## we could save a statue of the tracks to backup and then
## reset will get that copy back
xlim(obj) <- c(6145000, 6150000)
obj <- backup(obj)
obj@xlim <- c(6135000, 6150000)
obj
obj <- reset(obj)
obj

## @knitr utils
## ============================================================
## utils
## ============================================================
## summary information about a track
summary(obj)
## update a x limits on the fly, this is useful when you try to
## keep the view open and tweak with limits on the fly.
update(obj, xlim  = c(6130000, 6150000))

## @knitr opts
## ============================================================
## options
## ============================================================
## To make it easy, you could just apply any *options* by using "+"
## and this will apply it to every plot in the track.
obj + theme_bw()
```

Index