

Package ‘trackViewer’

October 16, 2019

Type Package

Title A R/Bioconductor package with web interface for drawing elegant interactive tracks or lollipop plot to facilitate integrated analysis of multi-omics data

Version 1.20.5

Author@R c(person(given="`Jianhong", family="`Ou", email="`jianhong.ou@duke.edu", role=c("`aut", "`cre"), comment=c(ORCID="`0000-0002-8652-2488")), person(given="`Julie", family="`Zhu", middle="`Lihua", role="`aut", email="`Julie.Zhu@umassmed.edu"))

Author Jianhong Ou and Lihua Julie Zhu

Maintainer Jianhong Ou <jianhong.ou@duke.edu>

Description Visualize mapped reads along with annotation as track layers for NGS dataset such as CHIP-seq, RNA-seq, miRNA-seq, DNA-seq, SNPs and methylation data.

License GPL (>= 2)

Depends R (>= 3.1.0), grDevices, methods, GenomicRanges, grid

Imports GenomeInfoDb, GenomicAlignments, GenomicFeatures, Gviz, Rsamtools, S4Vectors, rtracklayer, BiocGenerics, scales, tools, IRanges, AnnotationDbi, grImport, htmlwidgets, plotrix, Rgraphviz, InteractionSet, graph, utils

Suggests biomaRt, TxDb.Hsapiens.UCSC.hg19.knownGene, RUnit, org.Hs.eg.db, BiocStyle, knitr, VariantAnnotation

biocViews Visualization

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

git_url <https://git.bioconductor.org/packages/trackViewer>

git_branch RELEASE_3_9

git_last_commit b845362

git_last_commit_date 2019-08-07

Date/Publication 2019-10-15

R topics documented:

trackViewer-package	2
addArrowMark	3
addGuideLine	4
browseTracks	5
browseTracks-shiny	6
coverageGR	6
dandelion.plot	7
geneModelFromTxdb	8
geneTrack	9
getCurTrackViewport	9
getLocation	10
gieStain	11
gridPlot	11
GRoperator	12
ideogramPlot	12
importBam	14
importData	14
importScore	15
loadIdeogram	16
lollipop	17
optimizeStyle	18
parse2GRanges	19
parseWIG	19
plotGInteractions	20
plotGRanges	21
plotIdeo	22
plotOneIdeo	22
pos-class	23
trackList-class	24
trackStyle-class	24
trackViewerStyle-class	26
viewGene	27
viewTracks	28
xscale-class	29
yaxisStyle-class	29
Index	30

trackViewer-package *Minimal designed plotting tool for genomic data*

Description

A package that plot data and annotation information along genomic coordinates in an elegance style. This tool is based on Gviz but want to draw figures in minimal style for publication.

Examples

```

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
trs <- geneModelFromTxdb(TxDb.Hsapiens.UCSC.hg19.knownGene,
                        org.Hs.eg.db,
                        chrom="chr11",
                        start=122929275,
                        end=122930122)
extdata <- system.file("extdata", package="trackViewer",
                      mustWork=TRUE)
repA <- importScore(paste(extdata, "cpsf160.repA+.wig", sep="/"),
                  paste(extdata, "cpsf160.repA-.wig", sep="/"),
                  format="WIG")
strand(repA@dat) <- "+"
strand(repA@dat2) <- "-"
fox2 <- importScore(paste(extdata, "fox2.bed", sep="/"), format="BED")
dat <- coverageGR(fox2@dat)
fox2@dat <- dat[strand(dat)=="+"]
fox2@dat2 <- dat[strand(dat)=="-"]
gr <- GRanges("chr11", IRanges(122929275, 122930122), strand="-")
vp <- viewTracks(trackList(repA, fox2, trs), gr=gr, autoOptimizeStyle=TRUE)
addGuideLine(c(122929767, 122929969), vp=vp)
addArrowMark(list(x=unit(.5, "npc"),
                 y=unit(.39, "npc"),
                 col="blue")

```

addArrowMark

Add arrow mark to the figure at a given position

Description

A function to add arrow mark for emphasizing peaks

Usage

```

addArrowMark(pos = grid.locator(), label = NULL, angle = 15,
            length = unit(0.25, "inches"), col = "red", cex = 1,
            quadrant = 4, type = "closed", vp = NULL)

```

Arguments

pos	A unit object representing the location of arrow mark to be placed at current viewport. Default is the value of grid.locator, which will get the location of the mouse click.
label	A character or expression vector.
angle	A parameter passed into grid::arrow function. The angle of arrow head in degrees (smaller numbers produce narrower, pointier arrows). Essentially describes the width of the arrow head.
length	A parameter passed into grid::arrow function. A unit specifying the length of the arrow head.
col	color of the arrow

cex	Multiplier applied to fontsize
quadrant	the direction of arrow, 1: to bottomleft, 2: to bottomright, 3: to topright, 4: to topleft
type	A parameter passed into grid::arrow function. One of "open" or "closed" indicating whether the arrow head should be a closed triangle.
vp	A Grid viewport object. It must be output of viewTracks

Value

invisible x, y position value.

See Also

See Also as [addGuideLine](#), [arrow](#)

Examples

```
grid.newpage()
addArrowMark(list(x=unit(.5, "npc"),
                 y=unit(.5, "npc"),
                 label="label1",
                 col="blue")
## how to get the position by mouse click
if(interactive()){
pos <- addArrowMark(label="byClick")
addArrowMark(pos, label="samePosAsAbove")
}
```

addGuideLine	<i>Add guide lines to the tracks</i>
--------------	--------------------------------------

Description

A function to add lines for emphasizing the positions

Usage

```
addGuideLine(guideLine, col = "gray", lty = "dashed", lwd = 1,
             vp = NULL)
```

Arguments

guideLine	The genomic coordinates to draw the lines
col	A vector for the line color
lty	A vector for the line type
lwd	A vector for the line width
vp	A Grid viewport object. It must be output of viewTracks

See Also

See Also as [getCurTrackViewport](#), [addArrowMark](#), [viewTracks](#)

Examples

```
vp <- getCurTrackViewport(trackViewerStyle(), 10000, 10200)
addGuideLine(c(10010, 10025, 10150), vp=vp)
```

browseTracks	<i>browse tracks</i>
--------------	----------------------

Description

browse tracks by a web browser.

Usage

```
browseTracks(trackList, gr = GRanges(), ignore.strand = TRUE,
             width = NULL, height = NULL, ...)
```

Arguments

trackList	an object of <code>trackList</code>
gr	an object of <code>GRanges</code>
ignore.strand	ignore the strand or not when do filter. default TRUE
width	width of the figure
height	height of the figure
...	parameters not used

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

Examples

```
extdata <- system.file("extdata", package="trackViewer", mustWork=TRUE)
files <- dir(extdata, "-.wig")
tracks <- lapply(paste(extdata, files, sep="/"),
                importScore, format="WIG")
tracks <- lapply(tracks, function(.ele) {strand(.ele@dat) <- "-"; .ele})
names(tracks) <- c("trackA", "trackB")
fox2 <- importScore(paste(extdata, "fox2.bed", sep="/"), format="BED")
dat <- coverageGR(fox2@dat)
fox2@dat <- dat[strand(dat)=="+"]
fox2@dat2 <- dat[strand(dat)=="-"]
gr <- GRanges("chr11", IRanges(122929275, 122930122))
browseTracks(trackList(tracks, fox2), gr=gr)
```

browseTracks-shiny *Shiny bindings for browseTracks*

Description

Output and render functions for using browseTracks within Shiny applications and interactive Rmd documents.

Usage

```
browseTracksOutput(outputId, width = "100%", height = "600px")
```

```
renderbrowseTracks(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a browseTracks
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

coverageGR *calculate coverage*

Description

calculate coverage for [GRanges](#), [GAlignments](#) or [GAlignmentPairs](#)

Usage

```
coverageGR(gr)
```

Arguments

gr	an object of RGanges , GAlignments or GAlignmentPairs
----	---

Value

an object of [GRanges](#)

See Also

See Also as [coverage](#), [coverage-methods](#)

Examples

```
bed <- system.file("extdata", "fox2.bed", package="trackViewer",
                  mustWork=TRUE)
fox2 <- importScore(bed)
fox2$dat <- coverageGR(fox2$dat)
```

dandelion.plot

dandelion.plots

Description

Plot variants and somatic mutations

Usage

```
dandelion.plot(SNP.gr, features = NULL, ranges = NULL,
              type = c("fan", "circle", "pie", "pin"), newpage = TRUE,
              ylab = TRUE, ylab.gp = gpar(col = "black"), xaxis = TRUE,
              xaxis.gp = gpar(col = "black"), yaxis = FALSE, yaxis.gp = gpar(col
              = "black"), legend = NULL, cex = 1, maxgaps = 1/50,
              heightMethod = NULL, ...)
```

Arguments

SNP.gr	A object of GRanges or GRangesList . All the width of GRanges must be 1.
features	A object of GRanges or GRangesList .
ranges	A object of GRanges or GRangesList .
type	Character. Could be fan, circle, pie or pin.
newpage	plot in the new page or not.
ylab	plot ylab or not. If it is a character vector, the vector will be used as ylab.
ylab.gp, xaxis.gp, yaxis.gp	An object of class gpar for ylab, xaxis or yaxis.
xaxis, yaxis	plot xaxis/yaxis or not. If it is a numeric vector with length greater than 1, the vector will be used as the points at which tick-marks are to be drawn. And the names of the vector will be used to as labels to be placed at the tick points if it has names.
legend	If it is a list with named color vectors, a legend will be added.
cex	cex will control the size of circle.
maxgaps	maxgaps between the stem of dandelions. It is calculated by the width of plot region divided by maxgaps. If a GRanges object is set, the dandelions stem will be clustered in each genomic range.
heightMethod	A function used to determine the height of stem of dandelion. eg. Mean. Default is length.
...	not used.

Details

In `SNP.gr` and `features`, metadata of the `GRanges` object will be used to control the color, fill, border, height, data source of pie if the type is pie.

Examples

```
SNP <- c(10, 100, 105, 108, 400, 410, 420, 600, 700, 805, 840, 1400, 1402)
SNP.gr <- GRanges("chr1", IRanges(SNP, width=1, names=paste0("snp", SNP)),
                  score=sample.int(100, length(SNP))/100)
features <- GRanges("chr1", IRanges(c(1, 501, 1001),
                                   width=c(120, 500, 405),
                                   names=paste0("block", 1:3)),
                  color="black",
                  fill=c("#FF8833", "#51C6E6", "#DFA32D"),
                  height=c(0.1, 0.05, 0.08))
dandelion.plot(SNP.gr, features, type="fan")
```

geneModelFromTxdb *Prepare gene model from an object of TxDb*

Description

Generate an object of `track` for `viewTracks` by given parameters.

Usage

```
geneModelFromTxdb(txdb, orgDb, gr, chrom, start, end, strand = c("*",
"+", "-"), txdump = NULL)
```

Arguments

<code>txdb</code>	An object of <code>TxDb</code>
<code>orgDb</code>	An object of "OrgDb"
<code>gr</code>	An object of <code>GRanges</code> .
<code>chrom</code>	chromosome name, must be a seqname of <code>txdb</code>
<code>start</code>	start position
<code>end</code>	end position
<code>strand</code>	strand
<code>txdump</code>	output of <code>as.list(txdb)</code> , a list of data frames that can be used to make the db again with no loss of information.

Value

An object of `track`

See Also

See Also as `importScore`, `importBam`, `viewTracks`

Examples

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
trs <- geneModelFromTxdb(TxDb.Hsapiens.UCSC.hg19.knownGene,
                        org.Hs.eg.db,
                        chrom="chr20",
                        start=22560000,
                        end=22565000,
                        strand="-")
```

geneTrack

track from TxDb

Description

Generate a track object from TxDb by given gene ids

Usage

```
geneTrack(ids, txdb, type = c("gene", "transcript"))
```

Arguments

ids	Gene IDs. A vector of character. It should be keys in txdb.
txdb	An object of TxDb
type	Output type of track, "gene" or "transcript".

Value

An object of [track](#)

Examples

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
geneTrack(c("3312", "3313"), TxDb.Hsapiens.UCSC.hg19.knownGene)
```

getCurTrackViewport

Get current track viewport

Description

Get current track viewport for addGuideLine

Usage

```
getCurTrackViewport(curViewerStyle, start, end)
```

Arguments

curViewerStyle an object of [trackViewerStyle](#)
start start position of current track
end end position of current track

Value

an object of [viewport](#)

See Also

See Also as [addGuideLine](#)

Examples

```
vp <- getCurTrackViewport(trackViewerStyle(), 10000, 10200)
addGuideLine(c(10010, 10025, 10150), vp=vp)
```

getLocation

get genomic location by gene symbol

Description

given a gene name, get the genomic coordinates.

Usage

```
getLocation(symbol, txdb, org)
```

Arguments

symbol Gene symbol
txdb txdb will be used to extract the genes
org org package name

Examples

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
getLocation("HSPA8", TxDb.Hsapiens.UCSC.hg19.knownGene, "org.Hs.eg.db")
```

gieStain	<i>color scheme for the schema for Chromosome Band (Ideogram)</i>
----------	---

Description

Describe the colors of giemsa stain results

Usage

```
gieStain()
```

Value

A character vector of colors

Examples

```
gieStain()
```

gridPlot	<i>plot GRanges metadata</i>
----------	------------------------------

Description

plot GRanges metadata for different types

Usage

```
gridPlot(gr, gp, type, xscale)
```

Arguments

gr	an object of GRanges with metadata. All metadata must be numeric.
gp	an object of gpar
type	type of the figure, could be barplot, line, point and heatmap
xscale	x scale of the viewport

 GRoperator

GRanges operator

Description

GRanges operations (add, subtract, multiply, divide)

Usage

```
GRoperator(A, B, col = "score", operator = c("+", "-", "*", "/", "^",
"%"), ignore.strand = TRUE)
```

Arguments

A	an object of GRanges
B	an object of GRanges
col	colname of A and B to be calculated
operator	operator, "+" means A + B, and so on. User-defined function also could be used.
ignore.strand	When set to TRUE, the strand information is ignored in the overlap calculations.

Value

an object of GRanges

Examples

```
gr2 <- GRanges(seqnames=c("chr1", "chr1"),
  ranges=IRanges(c(7,13), width=3),
  strand=c("-", "-"), score=3:4)
gr3 <- GRanges(seqnames=c("chr1", "chr1"),
  ranges=IRanges(c(1, 4), c(3, 9)),
  strand=c("-", "-"), score=c(6L, 2L))
GRoperator(gr2, gr3, col="score", operator="+")
GRoperator(gr2, gr3, col="score", operator="-")
GRoperator(gr2, gr3, col="score", operator="*")
GRoperator(gr2, gr3, col="score", operator="/")
GRoperator(gr2, gr3, col="score", operator=mean)
```

 ideogramPlot

plot ideogram with data

Description

plot ideogram with data for multiple chromosomes

Usage

```
ideogramPlot(ideo, dataList, layout = NULL, horiz = TRUE,
  parameterList = list(vp = plotViewport(margins = c(0.1, 4.1, 0.3,
    0.1)), ideoHeight = unit(1/(1 + length(dataList)), "npc"), vgap =
    unit(0.3, "lines"), ylabs = "auto", ylabsRot = ifelse(horiz, 0, 90),
    ylabsPos = unit(2.5, "lines"), xaxis = FALSE, yaxis = FALSE, xlab = "",
    types = "barplot", heights = NULL, dataColumn = "score", gps = gpar(col =
    "black", fill = "gray")), colorScheme = gieStain(), gp = gpar(fill =
    NA, lwd = 2), ...)
```

Arguments

ideo	output of loadIdeogram .
dataList	a GRangesList of data to plot.
layout	The layout of chromosomes. Could be a list with chromosome names as its elements.
horiz	a logical value. If FALSE, the ideograms are drawn vertically to the left. If TRUE, the ideograms are drawn horizontally at the bottom.
parameterList	a list of parameters for each dataset in the dataList. The elements of the parameters could be xlabs, ylabs, etc. type could be barplot, line, point, heatmap.
colorScheme	A character vector of giemsa stain colors.
gp	parameters used for grid.roundrect .
...	parameters not used.

Examples

```
## Not run:
ideo <- loadIdeogram("hg38")
library(rtracklayer)
library(grid)
dataList <- ideo
dataList$score <- as.numeric(dataList$gieStain)
dataList <- dataList[dataList$gieStain!="gneg"]
dataList <- GRangesList(dataList)
grid.newpage()
ideogramPlot(ideo, dataList,
  layout=list("chr1", "chr2", c("chr3", "chr22"),
    c("chr4", "chr21"), c("chr5", "chr20"),
    c("chr6", "chr19"), c("chr7", "chr18"),
    c("chr8", "chr17"), c("chr9", "chr16"),
    c("chr10", "chr15"), c("chr11", "chr14"),
    c("chr12", "chr13"), c("chrX", "chrY")),
  parameterList = list(types="heatmap", colorKeyTitle="sample1"))

## End(Not run)
```

importBam	<i>Reading data from a BAM file</i>
-----------	-------------------------------------

Description

Read a [track](#) object from a BAM file

Usage

```
importBam(file, file2, ranges = GRanges(), pairs = FALSE)
```

Arguments

file	The path to the BAM file to read.
file2	The path to the second BAM file to read.
ranges	An object of GRanges to indicate the range to be imported
pairs	logical object to indicate the BAM is paired or not. See readGAlignments

Value

a [track](#) object

See Also

See Also as [importScore](#), [track](#), [viewTracks](#)

Examples

```
bamfile <- system.file("extdata", "ex1.bam", package="Rsamtools",
mustWork=TRUE)
dat <- importBam(file=bamfile, ranges=GRanges("seq1", IRanges(1, 50), strand="+"))
```

importData	<i>Reading data from a BED or WIG file to RleList</i>
------------	---

Description

Read a [track](#) object from a BED, bedGraph, WIG or BigWig file to RleList

Usage

```
importData(files, format = NA, ranges = GRanges())
```

Arguments

files	The path to the files to read.
format	The format of import file. Could be BAM, BED, bedGraph, WIG or BigWig
ranges	An object of GRanges to indicate the range to be imported

Value

a list of [RleList](#).

Examples

```
#import a BED file
bedfile <- system.file("tests", "test.bed", package="rtracklayer",
  mustWork=TRUE)
dat <- importData(files=bedfile, format="BED",
  ranges=GRanges("chr7", IRanges(127471197, 127474697)))

##import a WIG file
wigfile <- system.file("tests", "step.wig", package = "rtracklayer",
  mustWork=TRUE)
dat <- importData(files=wigfile, format="WIG",
  ranges=GRanges("chr19",
    IRanges(59104701, 59110920)))

##import a BigWig file
if(!.Platform$OS.type!="windows"){
  ##this is because we are using rtracklayer::import
  bwfile <- system.file("tests", "test.bw", package = "rtracklayer",
    mustWork=TRUE)
  dat <- importData(files=bwfile, format="BigWig",
    ranges=GRanges("chr19", IRanges(1500, 2700)))
}
```

importScore

Reading data from a BED or WIG file

Description

Read a [track](#) object from a BED, bedGraph, WIG or BigWig file

Usage

```
importScore(file, file2, format = c("BED", "bedGraph", "WIG", "BigWig"),
  ranges = GRanges(), ignore.strand = TRUE)
```

Arguments

file	The path to the file to read.
file2	The path to the second file to read.
format	The format of import file. Could be BED, bedGraph, WIG or BigWig
ranges	An object of GRanges to indicate the range to be imported
ignore.strand	ignore the strand or not when do filter. default TRUE

Value

a [track](#) object

See Also

See Also as [importBam](#), [track](#), [viewTracks](#)

Examples

```
#import a BED file
bedfile <- system.file("tests", "test.bed", package="rtracklayer",
  mustWork=TRUE)
dat <- importScore(file=bedfile, format="BED",
  ranges=GRanges("chr7", IRanges(127471197, 127474697)))

##import a WIG file
wigfile <- system.file("tests", "step.wig", package = "rtracklayer",
  mustWork=TRUE)
dat <- importScore(file=wigfile, format="WIG")

##import a BigWig file
if(.Platform$OS.type!="windows"){##this is because we are using rtracklayer::import
  bwfile <- system.file("tests", "test.bw", package = "rtracklayer",
    mustWork=TRUE)
  dat <- importScore(file=bwfile, format="BigWig")
}

##import 2 file
wigfile1 <- system.file("extdata", "cpsf160.repA+.wig", package="trackViewer",
  mustWork=TRUE)
wigfile2 <- system.file("extdata", "cpsf160.repA-.wig", package="trackViewer",
  mustWork=TRUE)
dat <- importScore(wigfile1, wigfile2, format="WIG",
  ranges=GRanges("chr11", IRanges(122817703, 122889073)))
```

loadIdeogram

load ideogram from UCSC

Description

Download ideogram table from UCSC

Usage

```
loadIdeogram(genome, chrom = NULL, ranges = NULL, ...)
```

Arguments

genome	Assembly name assigned by UCSC, such as hg38, mm10.
chrom	A character vector of chromosome names, or NULL.
ranges	A Ranges object with the intervals.
...	Additional arguments to pass to the GRanges constructor.

Value

A [GRanges](#) object.

See Also

See Also as [ideogramPlot](#)

Examples

```
## Not run:
head(loadIdeogram("hg38"))

## End(Not run)
```

lollipoplot

Lollipoplots

Description

Plot variants and somatic mutations

Usage

```
lollipoplot(SNP.gr, features = NULL, ranges = NULL, type = "circle",
  newpage = TRUE, ylab = TRUE, ylab.gp = gpar(col = "black"),
  yaxis = TRUE, yaxis.gp = gpar(col = "black"), xaxis = TRUE,
  xaxis.gp = gpar(col = "black"), legend = NULL, cex = 1,
  dashline.col = "gray80", jitter = c("node", "label"),
  rescale = FALSE, ...)
```

Arguments

SNP.gr	A object of GRanges , GRangesList or a list of GRanges . All the width of GRanges must be 1.
features	A object of GRanges , GRangesList or a list of GRanges . The metadata 'featureLayerID' are used for drawing features in different layers. See details in vignette.
ranges	A object of GRanges or GRangesList .
type	character. Could be circle, pie, pin, pie.stack or flag.
newpage	Plot in the new page or not.
ylab	Plot ylab or not. If it is a character vector, the vector will be used as ylab.
ylab.gp, xaxis.gp, yaxis.gp	An object of class gpar for ylab, xaxis or yaxis.
yaxis	Plot yaxis or not.
xaxis	Plot xaxis or not. If it is a numeric vector with length greater than 1, the vector will be used as the points at which tick-marks are to be drawn. And the names of the vector will be used to as labels to be placed at the tick points if it has names.
legend	If it is a list with named color vectors, a legend will be added.
cex	cex will control the size of circle.
dashline.col	color for the dashed line.

jitter	jitter the position of nodes or labels.
rescale	logical(1) or a dataframe with rescale from and to. Recalse the x-axis or not. if dataframe is used, colnames must be from.start, from.end, to.start, to.end.
...	not used.

Details

In SNP.gr and features, metadata of the GRanges object will be used to control the color, fill, border, height, cex, dashline.col, data source of pie if the type is pie. And also the controls for labels by name the metadata start as label.parameter.<properties> such as label.parameter.rot, label.parameter.gp. The parameter is used for [grid.text](#). The metadata 'featureLayerID' for features are used for drawing features in different layers. The metadata 'SNPsideID' for SNP.gr are used for determining the side of lollipops. And the 'SNPsideID' could only be 'top' or 'bottom'.

Examples

```
SNP <- c(10, 100, 105, 108, 400, 410, 420, 600, 700, 805, 840, 1400, 1402)
x <- sample.int(100, length(SNP))
SNP.gr <- GRanges("chr1", IRanges(SNP, width=1, names=paste0("snp", SNP)),
                  value1=x, value2=100-x)
SNP.gr$color <- rep(list(c("red", 'blue')), length(SNP))
SNP.gr$border <- sample.int(7, length(SNP), replace=TRUE)
features <- GRanges("chr1", IRanges(c(1, 501, 1001),
                                   width=c(120, 500, 405),
                                   names=paste0("block", 1:3)),
                    color="black",
                    fill=c("#FF8833", "#51C6E6", "#DFA32D"),
                    height=c(0.1, 0.05, 0.08),
                    label.parameter.rot=45)
lollipop(SNP.gr, features, type="pie")
```

optimizeStyle

Optimize the style of plot

Description

Automatic optimize the stlye of trackViewer

Usage

```
optimizeStyle(trackList, viewerStyle = trackViewerStyle(),
              theme = NULL)
```

Arguments

trackList	An object of trackList
viewerStyle	An object of trackViewerStyle
theme	A character string. Could be "bw", "col" or "safe".

Value

a list of a [trackList](#) and a [trackViewerStyle](#)

See Also

See Also as [viewTracks](#)

Examples

```
extdata <- system.file("extdata", package="trackViewer",
                      mustWork=TRUE)
files <- dir(extdata, ".wig")
tracks <- lapply(paste(extdata, files, sep="/"),
               importScore, format="WIG")
re <- optimizeStyle(trackList(tracks))
trackList <- re$tracks
viewerStyle <- re$style
```

parse2GRanges

parse text into GRanges

Description

parse text like "chr13:99,443,451-99,848,821:-" into GRanges

Usage

```
parse2GRanges(text)
```

Arguments

text character vector like "chr13:99,443,451-99,848,821:-" or "chr13:99,443,451-99,848,821"

Value

an object of [GRanges](#)

Examples

```
parse2GRanges("chr13:99,443,451-99,848,821:-")
```

parseWIG

convert WIG format track to BED format track

Description

convert WIG format track to BED format track for a given range

Usage

```
parseWIG(trackScore, chrom, from, to)
```

Arguments

trackScore	an object of track with WIG format
chrom	sequence name of the chromosome
from	start coordinate
to	end coordinate

Value

an object of [track](#)

Examples

```
extdata <- system.file("extdata", package="trackViewer", mustWork=TRUE)
repA <- importScore(file.path(extdata, "cpsf160.repA_-.wig"),
                   file.path(extdata, "cpsf160.repA_+.wig"),
                   format="WIG")
strand(repA$dat) <- "-"
strand(repA$dat2) <- "+"
parseWIG(repA, chrom="chr11", from=122929275, to=122930122)
```

plotGInteractions *plot GInteractions*

Description

plot graph for GInteractions

Usage

```
plotGInteractions(gi, range, feature.gr, ...)
```

Arguments

gi	an object of GInteractions
range	the region to plot. an object of GRanges
feature.gr	the feature.gr to be added. an object of GRanges
...	Not used.

Examples

```
library(InteractionSet)
gi <- readRDS(system.file("extdata", "gi.rds", package="trackViewer"))
range <- GRanges("chr2", IRanges(234500000, 235000000))
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
feature.gr <- genes(TxDb.Hsapiens.UCSC.hg19.knownGene)
feature.gr <- subsetByOverlaps(feature.gr, regions(gi))
feature.gr$col <- sample(1:7, length(feature.gr), replace=TRUE)
feature.gr$type <- sample(c("promoter", "enhancer", "gene"),
                        length(feature.gr), replace=TRUE,
                        prob=c(0.1, 0.2, 0.7))
plotGInteractions(gi, range, feature.gr)
```

plotGRanges	<i>plot GRanges data</i>
-------------	--------------------------

Description

A function to plot GRanges data for given range

Usage

```
plotGRanges(..., range = GRanges(), viewerStyle = trackViewerStyle(),
  autoOptimizeStyle = FALSE, newpage = TRUE)
```

Arguments

...	one or more objects of GRanges
range	an object of GRanges
viewerStyle	an object of trackViewerStyle
autoOptimizeStyle	should use optimizeStyle to optimize style
newpage	should be draw on a new page?

Value

An object of [viewport](#) for [addGuideLine](#)

See Also

See Also as [addGuideLine](#), [addArrowMark](#)

Examples

```
gr1 <- GRanges("chr1", IRanges(1:50, 51:100))
gr2 <- GRanges("chr1", IRanges(seq(from=10, to=80, by=5),
  seq(from=20, to=90, by=5)))
vp <- plotGRanges(gr1, gr2, range=GRanges("chr1", IRanges(1, 100)))
addGuideLine(guideLine=c(5, 10, 50, 90), col=2:5, vp=vp)

gr <- GRanges("chr1", IRanges(c(1, 11, 21, 31), width=9),
  score=c(5, 10, 5, 1))
plotGRanges(gr, range=GRanges("chr1", IRanges(1, 50)))
```

plotIdeo	<i>plot ideogram</i>
----------	----------------------

Description

plot ideogram for one chromosome

Usage

```
plotIdeo(ideo, chrom = seqlevels(ideo)[1], colorScheme = gieStain(),
         gp = gpar(fill = NA), ...)
```

Arguments

ideo	output of loadIdeogram .
chrom	A length 1 character vector of chromosome name.
colorScheme	A character vector of giemsa stain colors.
gp	parameters used for grid.roundrect .
...	parameters not used.

Examples

```
## Not run:
ideo <- loadIdeogram("hg38")
library(grid)
grid.newpage()
plotIdeo(ideo)

## End(Not run)
```

plotOneIdeo	<i>plot ideogram with data for one chromosome</i>
-------------	---

Description

plot ideogram with data for one chromosome

Usage

```
plotOneIdeo(ideo, dataList, parameterList = list(vp =
  plotViewport(margins = c(0.1, 4.1, 1.1, 0.1)), ideoHeight = unit(1/(1 +
  length(dataList)), "npc"), vgap = unit(1, "lines"), ylabs =
  seqlevels(ideo)[1], ylabsRot = 90, ylabsPos = unit(2.5, "lines"), xaxis =
  FALSE, yaxis = FALSE, xlab = "", types = "barplot", heights = NULL,
  dataColumn = "score", gps = gpar(col = "black", fill = "gray")),
  chrom = seqlevels(ideo)[1], colorScheme = gieStain(), gp = gpar(fill
  = NA, lwd = 2), ...)
```

Arguments

ideo	output of loadIdeogram .
dataList	a GRangesList of data to plot.
parameterList	a list of parameters for each dataset in the dataList. The elements of the parameters could be xlabs, ylabs, etc. type could be barplot, line, point, heatmap.
chrom	A length 1 character vector of chromosome name.
colorScheme	A character vector of giemsa stain colors.
gp	parameters used for grid.roundrect .
...	parameters not used.

Examples

```
## Not run:
ideo <- loadIdeogram("hg38")
library(rtracklayer)
library(grid)
dataList <- ideo[seqnames(ideo) %in% "chr1"]
dataList$score <- as.numeric(dataList$gieStain)
dataList <- dataList[dataList$gieStain!="gneg"]
dataList <- GRangesList(dataList, dataList)
grid.newpage()
plotOneIdeo(ideo, dataList, chrom="chr1")

## End(Not run)
```

pos-class

Class "pos"

Description

An object of class "pos" represents a point location

Slots

x A [numeric](#) value, indicates the x position

y A [numeric](#) value, indicates the y position

unit "character" apesifying the units for the corresponding numeric values. See [unit](#)

trackList-class	<i>List of tracks</i>
-----------------	-----------------------

Description

An extension of List that holds only [track](#) objects.

Usage

```
trackList(..., heightDist = NA)
```

Arguments

...	Each tracks in ... becomes an element in the new trackList, in the same order. This is analogous to the list constructor, except every argument in ... must be derived from track .
heightDist	A vector or NA to define the height of each track.

See Also

[track](#).

trackStyle-class	<i>Class "trackStyle"</i>
------------------	---------------------------

Description

An object of class "trackStyle" represents track style.

An object of class "track" represents scores of a given track.

Usage

```
## S4 method for signature 'track'
show(object)

## S4 method for signature 'track'
x$name

## S4 replacement method for signature 'track'
x$name <- value

setTrackStyleParam(ts, attr, value)

## S4 method for signature 'track,character'
setTrackStyleParam(ts, attr, value)

setTrackXscaleParam(ts, attr, value)

## S4 method for signature 'track,character'
```



```

setTrackXscaleParam(ts, attr, value)

setTrackYaxisParam(ts, attr, value)

## S4 method for signature 'track,character'
setTrackYaxisParam(ts, attr, value)

```

Arguments

object	an object of trackStyle.
x	an object of trackStyle
name	slot name of trackStyle
value	values to be assigned.
ts	An object of track.
attr	the name of slot of trackStyle object to be changed.

Slots

tracktype "character" track type, could be peak or cluster. Default is "peak". "cluster" is not supported yet. #' @slot color "character" track color. If the track has dat and dat2 slot, it should have two values.

height "numeric" track height. It should be a value between 0 and 1

marginTop "numeric" track top margin

marginBottom "numeric" track bottom margin

xscale object of [xscale](#), describe the details of x-scale

yaxis object of [yaxisStyle](#), describe the details of y-axis

ylim "numeric" y-axis range

ylabpos "character", ylable position, ylabpos should be 'left', 'right', 'topleft', 'bottomleft', 'topright', 'bottomright', 'abovebaseline' or 'underbaseline'. For gene type track, it also could be 'upstream' or 'downstream'

ylablas "numeric" y lable direction. It should be a integer 0-3. See [par:las](#)

ylabgp A "list" object, It will convert to an object of class [gpar](#). This is basically a list of graphical parameter settings of y-label.

dat Object of class [GRanges](#) the scores of a given track. It should contain score metadata.

dat2 Object of class [GRanges](#) the scores of a given track. It should contain score metadata. When dat2 and dat is paired, dat will be drawn as positive value where dat2 will be drawn as negative value (-1 * score)

type The type of track. It could be 'data', 'gene', 'transcript' or 'lollipopData'.

format The format of the input. It could be "BED", "bedGraph", "WIG", "BigWig" or "BAM"

style Object of class [trackStyle](#)

name unused yet

See Also

Please try to use [importScore](#) and [importBam](#) to generate the object.

Examples

```

extdata <- system.file("extdata", package="trackViewer",
mustWork=TRUE)
fox2 <- importScore(file.path(extdata, "fox2.bed"), format="BED")
setTrackStyleParam(fox2, "color", c("red","green"))
setTrackXscaleParam(fox2, "gp", list(cex=.5))
setTrackYaxisParam(fox2, "gp", list(col="blue"))
fox2$dat <- GRanges(score=numeric(0))

```

trackViewerStyle-class

Class "trackViewerStyle"

Description

An object of class "trackViewerStyle" represents track viewer style.

Usage

```

trackViewerStyle(...)

setTrackViewerStyleParam(tvs, attr, value)

## S4 method for signature 'trackViewerStyle,character'
setTrackViewerStyleParam(tvs, attr,
value)

```

Arguments

...	Each argument in ... becomes an slot in the new trackViewerStyle.
tvs	An object of trackViewerStyle.
attr	the name of slot to be changed.
value	values to be assigned.

Slots

margin "numeric", specify the bottom, left, top and right margin.

xlas "numeric", label direction of x-axis mark. It should be a integer 0-3. See [par:las](#)

xgp A "list", object, It will convert to an object of class [gpar](#). This is basically a list of graphical parameter settings of x-axis. For y-axis, see [yaxisStyle](#)

xaxis "logical", draw x-axis or not

autolas "logical" automatic determine y label direction

flip "logical" flip the x-axis or not, default FALSE

Examples

```

tvs <- trackViewerStyle()
setTrackViewerStyleParam(tvs, "xaxis", TRUE)

```

viewGene	<i>plot tracks based on gene name</i>
----------	---------------------------------------

Description

given a gene name, plot the tracks.

Usage

```
viewGene(symbol, filenames, format, txdb, org, upstream = 1000,  
          downstream = 1000, anchor = c("gene", "TSS"), plot = FALSE)
```

Arguments

symbol	Gene symbol
filenames	files used to generate tracks
format	file format used to generate tracks
txdb	txdb will be used to extract the genes
org	org package name
upstream	upstream from anchor
downstream	downstream from anchor
anchor	TSS, or gene
plot	plot the tracks or not.

Value

an invisible list of a [trackList](#), a [trackViewerStyle](#) and a [GRanges](#)

Examples

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)  
library(org.Hs.eg.db)  
extdata <- system.file("extdata", package="trackViewer", mustWork=TRUE)  
filename = file.path(extdata, "fox2.bed")  
optSty <- viewGene("HSPA8", filenames=filename, format="BED",  
                  txdb=TxDb.Hsapiens.UCSC.hg19.knownGene,  
                  org="org.Hs.eg.db")
```

viewTracks	<i>plot the tracks</i>
------------	------------------------

Description

A function to plot the data for given range

Usage

```
viewTracks(trackList, chromosome, start, end, strand, gr = GRanges(),
  ignore.strand = TRUE, viewerStyle = trackViewerStyle(),
  autoOptimizeStyle = FALSE, newpage = TRUE, operator = NULL,
  smooth = FALSE)
```

Arguments

trackList	an object of trackList
chromosome	chromosome
start	start position
end	end position
strand	strand
gr	an object of GRanges
ignore.strand	ignore the strand or not when do filter. default TRUE
viewerStyle	an object of trackViewerStyle
autoOptimizeStyle	should use optimizeStyle to optimize style
newpage	should be draw on a new page?
operator	operator, could be +, -, *, /, ^, %%. "-" means dat - dat2, and so on.
smooth	logical(1) or numeric(1). Smooth the curve or not. If it is numeric, eg n, mean of nearby n points will be used for plot.

Value

An object of [viewport](#) for [addGuideLine](#)

See Also

See Also as [addGuideLine](#), [addArrowMark](#)

Examples

```
extdata <- system.file("extdata", package="trackViewer",
  mustWork=TRUE)
files <- dir(extdata, "-.wig")
tracks <- lapply(paste(extdata, files, sep="/"),
  importScore, format="WIG")
tracks <- lapply(tracks, function(.ele) {strand(.ele@dat) <- "-"; .ele})
fox2 <- importScore(paste(extdata, "fox2.bed", sep="/"), format="BED")
dat <- coverageGR(fox2@dat)
```

```
fox2@dat <- dat[strand(dat)=="+"]
fox2@dat2 <- dat[strand(dat)=="-"]
gr <- GRanges("chr11", IRanges(122929275, 122930122), strand="-")
viewTracks(trackList(track=tracks, fox2=fox2), gr=gr, autoOptimizeStyle=TRUE)
```

xscale-class	<i>Class "xscale"</i>
--------------	-----------------------

Description

An object of class "xscale" represents x-scale style.

Slots

from A [pos](#) class, indicates the start point position of x-scale.

to A [pos](#) class, indicates the end point position of x-scale.

label "character" the label of x-scale

gp A "list" object, It will convert to an object of class [gpar](#). This is basically a list of graphical parameter settings of x-scale.

draw A "logical" value indicating whether the x-scale should be draw.

yaxisStyle-class	<i>Class "yaxisStyle"</i>
------------------	---------------------------

Description

An object of class "yaxisStyle" represents y-axis style.

Slots

at "numeric" vector of y-value locations for the tick marks

label "logical" value indicating whether to draw the labels on the tick marks.

gp A "list" object, It will convert to an object of class [gpar](#). This is basically a list of graphical parameter settings of y-axis.

draw A "logical" value indicating whether the y-axis should be draw.

main A "logical" value indicating whether the y-axis should be draw in left (TRUE) or right (FALSE).

Index

\$, track-method (trackStyle-class), 24
\$<-, track-method (trackStyle-class), 24

addArrowMark, 3, 4, 21, 28
addGuideLine, 4, 4, 10, 21, 28
arrow, 4

browseTracks, 5
browseTracks-shiny, 6
browseTracksOutput
 (browseTracks-shiny), 6

coverage, 6
coverageGR, 6

dandelion.plot, 7

GAlignmentPairs, 6
GAlignments, 6
geneModelFromTxdb, 8
geneTrack, 9
getCurTrackViewport, 4, 9
getLocation, 10
gieStain, 11
GInteractions, 20
gpar, 11, 25, 26, 29
GRanges, 5–7, 11, 14–17, 19–21, 25, 27, 28
GRangesList, 7, 13, 17, 23
grid.roundrect, 13, 22, 23
grid.text, 18
gridPlot, 11
GRoperator, 12

ideogramPlot, 12, 17
importBam, 8, 14, 16, 25
importData, 14
importScore, 8, 14, 15, 25

loadIdeogram, 13, 16, 22, 23
lollipop, 17

numeric, 23

optimizeStyle, 18, 21, 28

par, 25, 26

parse2GRanges, 19
parseWIG, 19
plotGInteractions, 20
plotGRanges, 21
plotIdeo, 22
plotOneIdeo, 22
pos, 29
pos (pos-class), 23
pos-class, 23

Ranges, 16
readGAlignments, 14
renderbrowseTracks
 (browseTracks-shiny), 6
RleList, 15

setTrackStyleParam (trackStyle-class),
 24
setTrackStyleParam, track, character, ANY-method
 (trackStyle-class), 24
setTrackStyleParam, track, character-method
 (trackStyle-class), 24
setTrackViewerStyleParam
 (trackViewerStyle-class), 26
setTrackViewerStyleParam, trackViewerStyle, character, ANY
 (trackViewerStyle-class), 26
setTrackViewerStyleParam, trackViewerStyle, character-method
 (trackViewerStyle-class), 26
setTrackXscaleParam (trackStyle-class),
 24
setTrackXscaleParam, track, character, ANY-method
 (trackStyle-class), 24
setTrackXscaleParam, track, character-method
 (trackStyle-class), 24
setTrackYaxisParam (trackStyle-class),
 24
setTrackYaxisParam, track, character, ANY-method
 (trackStyle-class), 24
setTrackYaxisParam, track, character-method
 (trackStyle-class), 24
show, track-method (trackStyle-class), 24

track, 8, 9, 14–16, 20, 24
track (trackStyle-class), 24

track-class (trackStyle-class), 24
trackList, 5, 18, 27, 28
trackList (trackList-class), 24
trackList-class, 24
trackStyle, 25
trackStyle (trackStyle-class), 24
trackStyle-class, 24
trackViewer (trackViewer-package), 2
trackViewer-package, 2
trackViewerStyle, 10, 18, 21, 27, 28
trackViewerStyle
 (trackViewerStyle-class), 26
trackViewerStyle-class, 26
TxDb, 8, 9

unit, 23

viewGene, 27
viewport, 10, 21, 28
viewTracks, 4, 8, 14, 16, 19, 28

xscale, 25
xscale (xscale-class), 29
xscale-class, 29

yaxisStyle, 25, 26
yaxisStyle (yaxisStyle-class), 29
yaxisStyle-class, 29