

# Package ‘trackViewer’

October 18, 2022

**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** 1.32.1

**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 (>= 3.5.0), grDevices, methods, GenomicRanges, grid, Rcpp

**Imports** GenomeInfoDb, GenomicAlignments, GenomicFeatures, Gviz, Rsamtools, S4Vectors, rtracklayer, BiocGenerics, scales, tools, IRanges, AnnotationDbi, grImport, htmlwidgets, plotrix, Rgraphviz, InteractionSet, graph, utils, rhdf5

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

**biocViews** Visualization

**VignetteBuilder** knitr

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**LinkingTo** Rcpp

**git\_url** <https://git.bioconductor.org/packages/trackViewer>

**git\_branch** RELEASE\_3\_15

**git\_last\_commit** 7878202

**git\_last\_commit\_date** 2022-05-04

**Date/Publication** 2022-10-18

**Author** Jianhong Ou [aut, cre] (<<https://orcid.org/0000-0002-8652-2488>>),  
Julie Lihua Zhu [aut]

**R topics documented:**

trackViewer-package . . . . .	3
addArrowMark . . . . .	4
addGuideLine . . . . .	5
browseTracks . . . . .	6
browseTracks-shiny . . . . .	7
coverageGR . . . . .	7
dandelion.plot . . . . .	8
geneModelFromTxdb . . . . .	9
geneTrack . . . . .	11
getCurTrackViewport . . . . .	11
getGeneIDsFromTxDb . . . . .	12
getLocation . . . . .	13
gi2track . . . . .	13
gieStain . . . . .	14
gridPlot . . . . .	14
GRoperator . . . . .	15
ideogramPlot . . . . .	16
importBam . . . . .	17
importData . . . . .	18
importGInteractions . . . . .	19
importScore . . . . .	20
importScSeqScore . . . . .	21
listChromosomes . . . . .	23
listResolutions . . . . .	23
loadIdeogram . . . . .	24
lollipop . . . . .	24
optimizeStyle . . . . .	26
parse2GRanges . . . . .	27
parseWIG . . . . .	28
plotGInteractions . . . . .	28
plotGRanges . . . . .	29
plotIdeo . . . . .	30
plotOneIdeo . . . . .	31
pos-class . . . . .	32
trackList-class . . . . .	32
trackStyle-class . . . . .	33
trackViewerStyle-class . . . . .	35
viewGene . . . . .	36
viewTracks . . . . .	37
xscale-class . . . . .	39
yaxisStyle-class . . . . .	39

---

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.

## Author(s)

**Maintainer:** Jianhong Ou <jianhong.ou@duke.edu> ([ORCID](#))

**Authors:**

- Julie Lihua Zhu <Julie.Zhu@umassmed.edu>

## 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 <code>grid.locator</code> , which will get the location of the mouse click.
label	A character or expression vector.
angle	A parameter passed into <code>grid::arrow</code> 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 <code>grid::arrow</code> 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 <code>grid::arrow</code> 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 <a href="#">viewTracks</a>

### 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

*Add guide lines to the tracks*

---

**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 <a href="#">viewTracks</a>

**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 <a href="#">trackList</a>
gr	an object of <a href="#">GRanges</a>
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 <a href="#">RGanges</a> , <a href="#">GAlignments</a> or <a href="#">GAlignmentPairs</a>
----	---

### 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,
  label_on_feature = FALSE,
  ...
)
```

**Arguments**

SNP.gr	A object of <a href="#">GRanges</a> or <a href="#">GRangesList</a> . All the width of GRanges must be 1.
features	A object of <a href="#">GRanges</a> or <a href="#">GRangesList</a> .
ranges	A object of <a href="#">GRanges</a> or <a href="#">GRangesList</a> .
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.
label_on_feature	Labels of the feature directly on them. Default FALSE.
...	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

txdb	An object of <a href="#">TxDb</a>
orgDb	An object of "OrgDb"
gr	An object of <a href="#">GRanges</a> .
chrom	chromosome name, must be a seqname of txdb
start	start position
end	end position
strand	strand
txdump	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

Generate a list of [track](#) from a TxDb object.

## 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	<i>track from TxDb</i>
-----------	------------------------

---

**Description**

Generate a track object from TxDb by given gene ids

**Usage**

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

**Arguments**

ids	Gene IDs. A vector of character. It should be keys in txdb.
txdb	An object of <a href="#">TxDb</a> .
symbols	symbol of genes.
type	Output type of track, "gene" or "transcript".
asList	Output a list of tracks or not. Default TRUE.

**Value**

An object of [track](#)

**Examples**

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)
ids <- c("3312", "390259", "341056", "79827")
symbols <- mget(ids, org.Hs.egSYMBOL)
geneTrack(ids, TxDb.Hsapiens.UCSC.hg19.knownGene, symbols)
```

---

getCurTrackViewport	<i>Get current track viewport</i>
---------------------	-----------------------------------

---

**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)
```

---

getGeneIDsFromTxDb      *get gene ids by genomic location*

---

**Description**

retrieve gene ids from txdb object by genomic location.

**Usage**

```
getGeneIDsFromTxDb(gr, txdb)
```

**Arguments**

gr GRanges object.  
txdb An object of [TxDb](#).

**Value**

A character vector of gene ids

**Examples**

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
gr <- parse2GRanges("chr11:122,830,799-123,116,707")
ids <- getGeneIDsFromTxDb(gr, TxDb.Hsapiens.UCSC.hg19.knownGene)
```

---

getLocation	<i>get genomic location by gene symbol</i>
-------------	--

---

**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")
```

---

gi2track	<i>convert GInteractions to track object</i>
----------	--

---

**Description**

Convert GInteractions object to track object

**Usage**

```
gi2track(gi, gi2)
```

**Arguments**

gi	an object of GInteractions
gi2	an object of GInteractions

**Value**

an track object

**Examples**

```
gi <- readRDS(system.file("extdata", "nij.chr6.51120000.53200000.gi.rds", package="trackViewer"))
gi2track(gi)
```

---

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 <a href="#">GRanges</a> with metadata. All metadata must be numeric.
gp	an object of <a href="#">gpar</a>
type	type of the figure, could be barplot, line, point and heatmap
xscale	x scale of the viewport

---

GRoperator	<i>GRanges operator</i>
------------	-------------------------

---

## 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 <a href="#">loadIdeogram</a> .
dataList	a <a href="#">GRangesList</a> 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 <a href="#">grid.roundrect</a> .
...	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

*Reading data from a BAM file*


---

## 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 <a href="#">GRanges</a> to indicate the range to be imported
pairs	logical object to indicate the BAM is paired or not. See <a href="#">readGAlignments</a>

## 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`*Reading data from a BED or WIG file to RleList*

---

### Description

Read a `track` object from a BED, bedGraph, WIG or BigWig file to RleList

### Usage

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

### Arguments

<code>files</code>	The path to the files to read.
<code>format</code>	The format of import file. Could be BAM, BED, bedGraph, WIG or BigWig
<code>ranges</code>	An object of <code>GRanges</code> 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)))
}
```

---

importGInteractions    *Reading data from a ginteractions, hic, cool, or validPairs file*

---

## Description

Read a [track](#) object from a ginteractions, hic, mcool, or validPairs file

## Usage

```
importGInteractions(  
  file,  
  format = c("ginteractions", "hic", "cool", "validPairs"),  
  ranges = GRanges(),  
  ignore.strand = TRUE,  
  out = c("track", "GInteractions"),  
  resolution = 1e+05,  
  unit = c("BP", "FRAG"),  
  normalization = c("NONE", "VC", "VC_SORT", "KR", "balanced"),  
  ...  
)
```

## Arguments

file	The path to the file to read.
format	The format of import file. Could be ginteractions, hic, cool or validPairs
ranges	An object of <a href="#">GRanges</a> to indicate the range to be imported
ignore.strand	ignore the strand or not when do filter. default TRUE
out	output format. Default is track. Possible values: track, GInteractions.
resolution	Resolutions for the interaction data.
unit	BP (base pair) or FRAG (fragment) (.hic file only).
normalization	Type of normalization, NONE, VC, VC_SORT or KR for .hic and NONE, balanced for .cool.
...	NOT used.

## Value

a [track](#) object

## See Also

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

## Examples

```
#import a ginteractions file
#gi <- system.file("extdata", "test.ginteractions.tsv", package="trackViewer",
#                 mustWork=TRUE)
#dat <- importGInteractions(file=gi, format="ginteractions",
#                            ranges=GRanges("chr7", IRanges(127471197, 127474697)))

##import a hic file
if(!.Platform$OS.type!="windows"){
hic <- system.file("extdata", "test_chr22.hic", package = "trackViewer",
                  mustWork=TRUE)
dat <- importGInteractions(file=hic, format="hic",
                          ranges=GRanges("22", IRanges(1500000, 100000000)))
}

##import a cool file
cool <- system.file("extdata", "test.mcool", package = "trackViewer",
                  mustWork=TRUE)
dat <- importGInteractions(file=cool, format="cool",
                          resolution = 2,
                          ranges=GRanges("chr1", IRanges(10, 28)))

##import a validPairs file
#validPairs <- system.file("extdata", "test.validPairs", package = "trackViewer",
#                          mustWork=TRUE)
#dat <- importGInteractions(file=validPairs, format="validPairs")
```

---

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)))
```

---

importScSeqScore            *plot tracks for single cell RNAseq*

---

**Description**

Plot single cell RNAseq data as heatmap track for Seurat object.

**Usage**

```
importScSeqScore(
  object,
  files,
  samplenames,
  ...,
  txdb,
  gene,
  id,
  idents,
  gr,
  color,
  withCoverageTrack = TRUE,
  flag = scanBamFlag(isSecondaryAlignment = FALSE, isUnmappedQuery = FALSE,
    isNotPassingQualityControls = FALSE, isSupplementaryAlignment = FALSE)
)
```

**Arguments**

object	Seurat object.
files	bam file to be scanned.
samplenames	sample names for files.
...	parameters used by <a href="#">readBAlignmentsList</a> or <a href="#">readGAlignments</a>
txdb	TxDb object for gene model.
gene	Gene name to plot. (row value)
id	The id of gene used in txdb.
idents	identity class to define the groups to plot. (column value)
gr	GRanges object to define the plotting region.
color	vector of colors used in heatmap.
withCoverageTrack	plot coverage track or not.
flag	An integer(2) vector used to filter reads based on their 'flag' entry.

**Examples**

```
## Not run:
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
test_file <- "https://github.com/10XGenomics/subset-bam/raw/master/test/test.bam"
trs <- importScSeqScore(files=test_file,
  txdb=TxDb.Hsapiens.UCSC.hg19.knownGene,
  id="653635", gene = "WASH7P")

## End(Not run)
```

---

listChromosomes	<i>List the available chromosome</i>
-----------------	--------------------------------------

---

**Description**

List the chromosomes available in the file.

**Usage**

```
listChromosomes(file, format = c("hic", "cool"))
```

**Arguments**

file	character(1). File name of .hic or .cool/.mcool/.scool
format	character(1). File format, "hic" or "cool".

**Examples**

```
hicfile <- system.file("extdata", "test_chr22.hic", package="trackViewer")
listChromosomes(hicfile)
coolfile <- system.file("extdata", "test.mcool", package="trackViewer")
listChromosomes(coolfile, format="cool")
```

---

listResolutions	<i>List the available resolutions</i>
-----------------	---------------------------------------

---

**Description**

List the resolutions available in the file.

**Usage**

```
listResolutions(file, format = c("hic", "cool"), unit = "BP")
```

**Arguments**

file	character(1). File name of .hic or .cool/.mcool/.scool
format	character(1). File format, "hic" or "cool".
unit	character(1). Resolution unit, "BP" or "FRAG". For hic file only.

**Examples**

```
hicfile <- system.file("extdata", "test_chr22.hic", package="trackViewer")
listResolutions(hicfile)
coolfile <- system.file("extdata", "test.mcool", package="trackViewer")
listResolutions(coolfile, format="cool")
```

---

loadIdeogram	<i>load ideogram from UCSC</i>
--------------	--------------------------------

---

**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 <a href="#">Ranges</a> object with the intervals.
...	Additional arguments to pass to the <a href="#">GRanges</a> constructor.

**Value**

A [GRanges](#) object.

**See Also**

See Also as [ideogramPlot](#)

**Examples**

```
## Not run:  
head(loadIdeogram("hg38", chrom = "chr1"))  
  
## End(Not run)
```

---

lollipop	<i>Lolliplots</i>
----------	-------------------

---

**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,
  label_on_feature = FALSE,
  ...
)

```

**Arguments**

SNP.gr	A object of <a href="#">GRanges</a> , <a href="#">GRangesList</a> or a list of <a href="#">GRanges</a> . All the width of <a href="#">GRanges</a> must be 1.
features	A object of <a href="#">GRanges</a> , <a href="#">GRangesList</a> or a list of <a href="#">GRanges</a> . The metadata 'featureLayerID' are used for drawing features in different layers. See details in vignette.
ranges	A object of <a href="#">GRanges</a> or <a href="#">GRangesList</a> .
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), character(1), numeric vector, or a dataframe with rescale from and to. Rescale the x-axis or not. if dataframe is used, colnames must be from.start, from.end, to.start, to.end. And the from scale must cover the whole plot region. The rescale parameter can be set as "exon" or "intron" to emphasize "exon" or "intron" region. The "exon" or "intron" can be followed with an integer e.g. "exon\_80", or "intron\_99". The integer indicates the total percentage of "exon" or "intron" region. Here "exon" indicates all regions in features. And "intron" indicates all flank regions of the features.

label\_on\_feature    Labels of the feature directly on them. Default FALSE.

...                not used.

### Details

In SNP.gr and features, metadata of the GRanges object will be used to control the color, fill, border, alpha, shape, 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.grp. 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	<i>convert WIG format track to BED format track</i>
----------	---

---

**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	<i>plot GInteractions</i>
-------------------	---------------------------

---

**Description**

plot graph for GInteractions

**Usage**

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

**Arguments**

gi	an object of <a href="#">GInteractions</a>
range	the region to plot. an object of <a href="#">GRanges</a>
feature.gr	the feature.gr to be added. an object of <a href="#">GRanges</a>
...	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

*plot GRanges data*


---

**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 <a href="#">GRanges</a>
range	an object of <a href="#">GRanges</a>
viewerStyle	an object of <a href="#">trackViewerStyle</a>
autoOptimizeStyle	should use <a href="#">optimizeStyle</a> 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

*plot ideogram*

---

**Description**

plot ideogram for one chromosome

**Usage**

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

**Arguments**

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

**Examples**

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

## End(Not run)
```

---

plotOneIdeo

*plot ideogram with data for one chromosome*


---

**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],
  colorSheme = gieStain(),
  gp = gpar(fill = NA, lwd = 2),
  ...
)
```

**Arguments**

ideo	output of <a href="#">loadIdeogram</a> .
dataList	a <a href="#">GRangesList</a> 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.
colorSheme	A character vector of giemsa stain colors.
gp	parameters used for <a href="#">grid.roundrect</a> .
...	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

*List of tracks*


---

**Description**

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

**Usage**

```
## S4 replacement method for signature 'trackList'
seqlevelsStyle(x) <- value

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



**Arguments**

x	trackList object.
value	values to be assigned.
...	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 <a href="#">track</a> .
heightDist	A vector or NA to define the height of each track.

**See Also**

[track](#).

---

trackStyle-class	Class "trackStyle"
------------------	--------------------

---

**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'
seqlevels(x)

## S4 method for signature 'track'
seqlevelsStyle(x)

## S4 replacement method for signature 'track'
seqlevelsStyle(x) <- value

## 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

x	an object of trackStyle
value	values to be assigned.
object	an object of trackStyle.
name	slot name of trackStyle
ts	An object of track.
attr	the name of slot of <a href="#">trackStyle</a> object to be changed.

### Details

The attr of setTrackXscaleParam could not only be a slot of xscale, but also be position. If the attr is set to position, value must be a list of x, y and label. For example setTrackXscaleParam(track, attr="position", value=list(x=122929675, y=4, label=500))

### Slots

tracktype "character" track type, could be peak or cluster. Default is "peak". "cluster" is not supported yet. For interaction data, it could be "heatmap" or "link".

color "character" track color. If the track has dat and dat2 slot, it should have two values.

NAcolor "character" NA color for interactionData.

breaks "numeric" breaks for color keys of interactionData.

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 ( $-1 * \text{score}$ )  
**type** The type of track. It could be 'data', 'gene', 'transcript', 'scSeq', 'lollipopData' or 'interactionData'.  
**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 *plot tracks based on gene name*

---

**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 <a href="#">trackList</a>
chromosome	chromosome
start	start position
end	end position
strand	strand
gr	an object of <a href="#">GRanges</a>
ignore.strand	ignore the strand or not when do filter. default TRUE
viewerStyle	an object of <a href="#">trackViewerStyle</a>
autoOptimizeStyle	should use <a href="#">optimizeStyle</a> 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(). Plot smooth curve or not. If it is numeric, eg n, mean of nearby n points will be used for plot. If it is numeric, the second number will be the color. Default color is 2 (red).

**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	Class "xscale"
--------------	----------------

---

**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	Class "yaxisStyle"
------------------	--------------------

---

**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), 33
- \$<-, track-method (trackStyle-class), 33
- addArrowMark, 4, 5, 30, 38
- addGuideLine, 5, 5, 12, 30, 38
- arrow, 5
- browseTracks, 6
- browseTracks-shiny, 7
- browseTracksOutput  
    (browseTracks-shiny), 7
- coverage, 8
- coverageGR, 7
- dandelion.plot, 8
- GAlignmentPairs, 7
- GAlignments, 7
- geneModelFromTxdb, 9
- geneTrack, 11
- getCurTrackViewport, 5, 11
- getGeneIDsFromTxDb, 12
- getLocation, 13
- gi2track, 13
- gieStain, 14
- GInteractions, 29
- gpar, 14, 34, 36, 39
- GRanges, 6–8, 14, 17–19, 21, 24, 25, 27, 29, 35, 37, 38
- GRangesList, 8, 16, 25, 31
- grid.roundrect, 16, 30, 31
- grid.text, 26
- gridPlot, 14
- GRoperator, 15
- ideogramPlot, 16, 24
- importBam, 10, 17, 19, 21, 35
- importData, 18
- importGInteractions, 19
- importScore, 10, 17, 20, 35
- importScSeqScore, 21
- listChromosomes, 23
- listResolutions, 23
- loadIdeogram, 16, 24, 30, 31
- lollipop, 24
- numeric, 32
- optimizeStyle, 26, 29, 38
- par, 34, 36
- parse2GRanges, 27
- parseWIG, 28
- plotGInteractions, 28
- plotGRanges, 29
- plotIdeo, 30
- plotOneIdeo, 31
- pos, 39
- pos (pos-class), 32
- pos-class, 32
- Ranges, 24
- readGAlignments, 17, 22
- readGAlignmentsList, 22
- renderbrowseTracks  
    (browseTracks-shiny), 7
- RleList, 18
- seqlevels, track-method  
    (trackStyle-class), 33
- seqlevelsStyle, track-method  
    (trackStyle-class), 33
- seqlevelsStyle<-, track-method  
    (trackStyle-class), 33
- seqlevelsStyle<-, trackList-method  
    (trackList-class), 32
- setTrackStyleParam (trackStyle-class), 33
- setTrackStyleParam, track, character, ANY-method  
    (trackStyle-class), 33



setTrackStyleParam, track, character-method  
(trackStyle-class), 33

setTrackViewerStyleParam  
(trackViewerStyle-class), 35

setTrackViewerStyleParam, trackViewerStyle, character, ANY-method  
(trackViewerStyle-class), 35

setTrackViewerStyleParam, trackViewerStyle, character-method  
(trackViewerStyle-class), 35

setTrackXscaleParam (trackStyle-class),  
33

setTrackXscaleParam, track, character, ANY-method  
(trackStyle-class), 33

setTrackXscaleParam, track, character-method  
(trackStyle-class), 33

setTrackYaxisParam (trackStyle-class),  
33

setTrackYaxisParam, track, character, ANY-method  
(trackStyle-class), 33

setTrackYaxisParam, track, character-method  
(trackStyle-class), 33

show, track-method (trackStyle-class), 33

track, 9–11, 17–21, 28, 32, 33

track (trackStyle-class), 33

track-class (trackStyle-class), 33

trackList, 6, 27, 37, 38

trackList (trackList-class), 32

trackList-class, 32

trackStyle, 34, 35

trackStyle (trackStyle-class), 33

trackStyle-class, 33

trackViewer (trackViewer-package), 3

trackViewer-package, 3

trackViewerStyle, 12, 27, 29, 37, 38

trackViewerStyle  
(trackViewerStyle-class), 35

trackViewerStyle-class, 35

TxDB, 10–12

unit, 32

viewGene, 36

viewport, 12, 30, 38

viewTracks, 4, 5, 9, 10, 17, 19, 21, 27, 37

xscale, 34

xscale (xscale-class), 39

xscale-class, 39

yaxisStyle, 34, 36

yaxisStyle (yaxisStyle-class), 39

yaxisStyle-class, 39