

Package: Gviz (via r-universe)

October 11, 2026

Title Plotting data and annotation information along genomic coordinates

Version 1.57.4

Description Genomic data analyses requires integrated visualization of known genomic information and new experimental data. Gviz uses the biomaRt and the rtracklayer packages to perform live annotation queries to Ensembl and UCSC and translates this to e.g. gene/transcript structures in viewports of the grid graphics package. This results in genomic information plotted together with your data.

LazyLoad yes

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.3), methods, S4Vectors (>= 0.9.25), IRanges (>= 1.99.18), GenomicRanges (>= 1.61.1), grid

Imports XVector (>= 0.5.7), rtracklayer (>= 1.69.1), lattice, RColorBrewer, biomaRt (>= 2.11.0), AnnotationDbi (>= 1.27.5), Biobase (>= 2.15.3), GenomicFeatures (>= 1.61.4), ensemblDb (>= 2.11.3), BSgenome (>= 1.77.1), Biostrings (>= 2.77.2), biovizBase (>= 1.13.8), Rsamtools (>= 2.25.1), latticeExtra (>= 0.6-26), matrixStats (>= 0.8.14), GenomicAlignments (>= 1.45.1), cigarillo, Seqinfo, GenomeInfoDb, BiocGenerics (>= 0.11.3), digest (>= 0.6.8), graphics, grDevices, stats, utils

Suggests BSgenome.Hsapiens.UCSC.hg38, xml2, BiocStyle, knitr, rmarkdown, testthat

biocViews Visualization, Microarray, Sequencing

URL <https://github.com/ivanek/Gviz>

BugReports <https://github.com/ivanek/Gviz/issues>

VignetteBuilder knitr

License Artistic-2.0

Collate 'utils.R' 'ImageMap-class.R' 'DisplayPars-class.R'
 'GdObject-class.R' 'ReferenceTrack-class.R'
 'SequenceTrack-class.R' 'RangeTrack-class.R'
 'StackedTrack-class.R' 'NumericTrack-class.R'
 'DataTrack-class.R' 'AlignmentsTrack-class.R'
 'AnnotationTrack-class.R' 'GeneRegionTrack-class.R'
 'BiomartGeneRegionTrack-class.R' 'CustomTrack-class.R'
 'GenomeAxisTrack-class.R' 'Gviz-defunct.R' 'Gviz-deprecated.R'
 'HighlightTrack-class.R' 'IdeogramTrack-class.R'
 'OverlayTrack-class.R' 'UcscTrack.R' 'collapsing.R'
 'datasets.R' 'exportTracks.R' 'grouping.R' 'plotTracks.R'
 'settings.R'

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-08-11 09:15:10 UTC

RemoteUrl <https://github.com/bioc/Gviz>

RemoteRef HEAD

RemoteSha 482799c06b17a225b2981638ab2bb00012f6b78c

Contents

AlignmentsTrack-class	3
availableDefaultMapping	10
BiomartGeneRegionTrack-class	12
collapsing	17
CustomTrack-class	18
datasets	20
DetailsAnnotationTrack-class	20
DisplayPars-class	22
exportTracks	25
GeneRegionTrack-class	26
grouping	36
Gviz-defunct	37
Gviz-deprecated	37
HighlightTrack-class	37
IdeogramTrack-class	41
NumericTrack-class	45
plotTracks	47
RangeTrack-class	50
SequenceBSgenomeTrack-class	56
SequenceDNAStringSetTrack-class	57
SequenceRNAStringSetTrack-class	57
SequenceTrack-class	58
settings	64

StackedTrack-class	90
UcscTrack	93

Index	97
--------------	-----------

AlignmentsTrack-class	<i>AlignmentsTrack class and methods</i>
-----------------------	--

Description

A class to represent short sequences that have been aligned to a reference genome as they are typically generated in next generation sequencing experiments.

Usage

```
## S4 method for signature 'AlignmentsTrack'
initialize(
  .Object,
  stackRanges = GRanges(),
  stacks = numeric(),
  sequences = DNASTringSet(),
  referenceSequence = NULL,
  ...
)

## S4 method for signature 'ReferenceAlignmentsTrack'
initialize(
  .Object,
  stream,
  reference,
  mapping = list(),
  args = list(),
  defaults = list(),
  stacks = numeric(),
  stackRanges = GRanges(),
  sequences = Biostrings::DNASTringSet(),
  referenceSequence = NULL,
  ...
)

AlignmentsTrack(
  range = NULL,
  start = NULL,
  end = NULL,
  width = NULL,
  strand,
  chromosome,
  genome,
```

```

    stacking = "squish",
    id,
    cigar,
    mapq,
    flag = scanBamFlag(isUnmappedQuery = FALSE),
    isize,
    groupid,
    status,
    md,
    seqs,
    name = "AlignmentsTrack",
    isPaired = TRUE,
    importFunction,
    referenceSequence,
    ...
)

## S4 method for signature 'AlignmentsTrack'
values(x)

## S4 replacement method for signature 'AlignmentsTrack'
chromosome(GdObject) <- value

## S4 method for signature 'AlignmentsTrack'
stacks(GdObject)

## S4 method for signature 'AlignmentsTrack'
setStacks(GdObject, ...)

## S4 method for signature 'AlignmentsTrack'
subset(x, from = NULL, to = NULL, stacks = FALSE, use.defaults = TRUE, ...)

## S4 method for signature 'ReferenceAlignmentsTrack'
subset(x, from, to, chromosome, ...)

## S4 method for signature 'AlignmentsTrack'
drawGrid(GdObject, from, to)

## S4 method for signature 'AlignmentsTrack'
drawAxis(GdObject, ...)

## S4 method for signature 'AlignmentsTrack'
drawGD(GdObject, minBase, maxBase, prepare = FALSE, subset = TRUE, ...)

## S4 method for signature 'AlignmentsTrack'
show(object)

## S4 method for signature 'ReferenceAlignmentsTrack'

```

```
show(object)
```

Arguments

.Object .Object

stackRanges stackRanges

stacks stacks

sequences sequences

referenceSequence

An optional [SequenceTrack](#) object containing the reference sequence against which the reads have been aligned. This is only needed when mismatch information has to be added to the plot (i.e., the showMismatch display parameter is TRUE) because this is normally not encoded in the BAM file. If not provided through this argument, the [plotTracks](#) function is smart enough to detect the presence of a [SequenceTrack](#) object in the track list and will use that as a reference sequence.

... Additional items which will all be interpreted as further display parameters. See [settings](#) and the "Display Parameters" section below for details.

stream stream

reference reference

mapping mapping

args args

defaults defaults

range An optional meta argument to handle the different input types. If range is missing, all the relevant information to create the object has to be provided as individual function arguments (see below).

The different input options for range are:

- A character **string**: the path to a BAM file containing the read alignments. This results in the instantiation of a `ReferenceAlignmentsTrack` object, but for the user this implementation detail should be of no concern.
- A `GRanges` **object**: the genomic ranges of the individual reads, along with the optional additional metadata columns `id`, `cigar`, `mapq`, `flag`, `isize`, `groupid`, `status`, `md` and `seqs` (see the individual function parameters below for details). Calling the constructor on a `GRanges` object without further arguments, e.g. `AlignmentsTrack(range = obj)`, is equivalent to calling the `coerce` method `as(obj, "AlignmentsTrack")`.
- An [IRanges](#) **object**: almost identical to the `GRanges` case, except that the chromosome and strand information, as well as all additional metadata, has to be provided via the separate `chromosome`, `strand`, `feature`, `group` or `id` arguments, since none of it can be encoded directly in an `IRanges` object. None of these inputs are mandatory; if not provided explicitly, the defaults `chromosome = NA` and `strand = "*"` are used.
- A `data.frame` **object**: must contain at least the two mandatory columns `start` and `end` with the range coordinates. It may also contain chromosome

and strand columns; if these are absent, the values are taken from the separate chromosome or strand arguments instead. In addition, the id, cigar, mapq, flag, isize, groupid, status, md and seqs data can be provided as additional columns, with the same default-value rules as above.

start, end, width	Integer vectors, giving the start and the end coordinates for the individual track items, or their width. Two of the three need to be specified, and have to be of equal length or of length one, in which case this single value will be recycled. Otherwise, the usual R recycling rules for vectors do not apply here.
strand	Character vector, the strand information for the reads. It may be provided in the form + for the Watson strand, - for the Crick strand or * for either one of the two. Needs to be of equal length as the provided genomic coordinates, or of length 1. Please note that paired reads need to be on opposite strands, and erroneous entries will result in casting of an error.
chromosome	The chromosome on which the track's genomic ranges are defined. A valid UCSC chromosome identifier if <code>options(ucscChromosomeNames=TRUE)</code> . Please note that in this case only syntactic checking takes place, i.e., the argument value needs to be an integer, numeric character or a character of the form <code>chrX</code> , where <code>X</code> may be any possible string. The user has to make sure that the respective chromosome is indeed defined for the track's genome. If not provided here, the constructor will try to construct the chromosome information based on the available inputs, and as a last resort will fall back to the value <code>chrNA</code> . Please note that by definition all objects in the Gviz package can only have a single active chromosome at a time (although internally the information for more than one chromosome may be present), and the user has to call the <code>chromosome<-replacement</code> method in order to change to a different active chromosome.
genome	The genome on which the track's ranges are defined. Usually this is a valid UCSC genome identifier, however this is not being formally checked at this point. If not provided here the constructor will try to extract this information from the provided input, and eventually will fall back to the default value of <code>NA</code> .
stacking	The stacking type for overlapping items of the track. One in <code>c(hide, dense, squish, pack, full)</code> . Currently, only <code>squish</code> (make best use of the available space), <code>dense</code> (no stacking, collapse overlapping ranges), and <code>hide</code> (do not show any track items at all) are implemented.
id	Character vector of read identifiers. Those identifiers have to be unique, i.e., each range representing a read needs to have a unique id.
cigar	A character vector of valid CIGAR strings describing details of the alignment. Typically those include alignment gaps or insertions and deletions, but also hard and soft clipped read regions. If missing, a fully mapped read without gaps or indels is assumed. Needs to be of equal length as the provided genomic coordinates, or of length 1.
mapq	A numeric vector of read mapping qualities. Needs to be of equal length as the provided genomic coordinates, or of length 1.
flag	A named integer vector of length 2, as is produced by <code>Rsamtools::scanBamFlag()</code> , used to filter out undesirable reads. If missing, all mapped reads will be included.

isize	A numeric vector of empirical insert sizes. This only applies if the reads are paired. Needs to be of equal length as the provided genomic coordinates, or of length 1. Currently not used.
groupid	A factor (or vector than can be coerced into one) defining the read pairs. Reads with the same groupid are considered to be mates. Please note that each read group may only have one or two members. Needs to be of equal length as the provided genomic coordinates, or of length 1.
status	A factor describing the mapping status of a read. Has to be one in mated, unmated or ambiguous. Needs to be of equal length as the provided genomic coordinates, or of length 1.
md	A character vector describing the mapping details. This is effectively and alternative to the CIGAR encoding and it removes the dependency on a reference sequence to figure out read mismatches. Needs to be of equal length as the provided genomic coordinates, or of length 1. Currently not used.
seqs	DNAStrngSet of read sequences.
name	Character scalar of the track's name used in the title panel when plotting.
isPaired	A logical scalar to determine whether the reads are paired or not. While this may be used to render paired-end data as single-end, the oppsite will typically not have any effect because the appropriate groupid settings will not be present. Thus setting isPaired to TRUE can usually be used to autotdetect the pairing state of the input data.
importFunction	A user-defined function to be used to import the data from a file. This only applies when the range argument is a character string with the path to the input data file. The function needs to accept an argument x containing the file path and a second argument selection with the desired plotting ranges. It has to return a proper GRanges object with all the necessary metadata columns set. A single default import function is already implemented in the package for BAM files.
x	x
GdObject	Object of GdObject-class.
value	value
from,to	from,to
use.defaults	logical
minBase, maxBase	minBase,maxBase
prepare	logical
subset	logical
object	object

Value

The return value of the constructor function is a new object of class `AlignmentsTrack` or `ReferenceAlignmentsTrack`.

Functions

- `initialize(AlignmentsTrack)`: Initialize the `stackRanges`, `stacks`, `sequences` and `referenceSequence` slots before deferring to the `StackedTrack` initializer for the remaining slots.
- `initialize(ReferenceAlignmentsTrack)`: Initialize the `ReferenceTrack` slots (`stream`, `reference`, `mapping`, `args`, `defaults`) and the `referenceSequence` slot inherited from `AlignmentsTrack`.
- `AlignmentsTrack()`: Constructor for `AlignmentsTrack`-class.
- `values(AlignmentsTrack)`: Return all additional annotation information except for the genomic coordinates for the track items as a `data.frame`.
- `chromosome(AlignmentsTrack) <- value`: replace the value of the track's chromosome. This has to be a valid UCSC chromosome identifier or an integer or character scalar that can be reasonably coerced into one.
- `stacks(AlignmentsTrack)`: return the stack indices for each track item.
- `setStacks(AlignmentsTrack)`: recompute the stacks based on the available space and on the object's track items and stacking settings.
- `subset(AlignmentsTrack)`: Subset an `AlignmentsTrack` by coordinates and sort if necessary.
- `subset(ReferenceAlignmentsTrack)`: Subset a `ReferenceAlignmentsTrack` by coordinates and sort if necessary.
- `drawGrid(AlignmentsTrack)`: superpose a horizontal grid on the coverage portion of the track, based on the `grid`, `h`, `col.grid`, `lty.grid` and `lwd.grid` display parameters.
- `drawAxis(AlignmentsTrack)`: add a y-axis to the title panel of a track.
- `drawGD(AlignmentsTrack)`: plot the object to a graphics device. The return value of this method is the input object, potentially updated during the plotting operation. Internally, there are two modes in which the method can be called. Either in 'prepare' mode, in which case no plotting is done but the object is preprocessed based on the available space, or in 'plotting' mode, in which case the actual graphical output is created. Since subsetting of the object can be potentially costly, this can be switched off in case subsetting has already been performed before or is not necessary.
- `show(AlignmentsTrack)`: Show method.
- `show(ReferenceAlignmentsTrack)`: Show method.

Objects from the Class

Objects can be created using the constructor function `AlignmentsTrack`.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)

[GdObject](#)

[GRanges](#)

[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## Creating objects
afrom <- 2960000
ato <- 3160000
alTrack <- AlignmentsTrack(system.file(
  package = "Gviz", "extdata",
  "gapped.bam"
), isPaired = TRUE)
plotTracks(alTrack, from = afrom, to = ato, chromosome = "chr12")

## Omit the coverage or the pile-ups part
plotTracks(alTrack,
  from = afrom, to = ato, chromosome = "chr12",
  type = "coverage"
)
plotTracks(alTrack,
  from = afrom, to = ato, chromosome = "chr12",
  type = "pileup"
)

## Including sequence information with the constructor
if (require(BSgenome.Hsapiens.UCSC.hg38)) {
  strack <- SequenceTrack(Hsapiens, chromosome = "chr21")
  afrom <- 44945200
  ato <- 44947200
  alTrack <- AlignmentsTrack(system.file(
    package = "Gviz", "extdata",
    "snps.bam"
  ), isPaired = TRUE, referenceSequence = strack)
  plotTracks(alTrack, chromosome = "chr21", from = afrom, to = ato)

  ## Including sequence information in the track list
  alTrack <- AlignmentsTrack(system.file(
    package = "Gviz", "extdata",
    "snps.bam"
  ), isPaired = TRUE)
  plotTracks(c(alTrack, strack),
```

```

        chromosome = "chr21", from = 44946590,
        to = 44946660
    )
}

```

availableDefaultMapping

ReferenceTrack class and methods

Description

A class allow for on-demand streaming of data off the file system.

Usage

```

availableDefaultMapping(file, trackType)

## S4 method for signature 'ReferenceTrack'
initialize(
  .Object,
  stream,
  reference,
  mapping = list(),
  args = list(),
  defaults = list()
)

```

Arguments

file	A character scalar with a file name or just a file extension.
trackType	A character scalar with one of the available track types in the package.
.Object	.Object
stream	stream
reference	reference
mapping	mapping
args	ars
defaults	defaults

Details

The availableDefaultMappings function can be used to find out whether the package defines a mapping scheme between one of the many supported input file types and the metadata columns of the tracks' GRanges objects.

Value

Constructor functions of AnnotationTrack, DataTrack, SequenceTrack and AlignmentsTrack can create a special object of corresponding Reference*Track subclass with pointer to the referenced file.

A virtual class: No objects may be created from it.

Functions

- availableDefaultMapping(): Function to find out whether the package defines a mapping scheme between one of the many supported input file types and the metadata columns of the tracks's GRanges objects.
- initialize(ReferenceTrack): Initialize all the ReferenceTrack slots (stream, reference, mapping, args, defaults) from the supplied arguments and validate the resulting object.

Slots

stream Object of class function. The import function to stream data of the file system. Needs to be able to handle the two mandatory arguments file (a character containing a valid file path) and selection (a GRanges object with the genomic region to plot).

reference Object of class "character", the path to the file containing the data.

mapping Object of class list, a default mapping between the metadata columns of the returned GRanges object from the import function and the elementMetadata columns that make up the final track object.

args Object of class list, the passed in constructor arguments during object instantiation. Those will be needed when fetching the data in order to fill all necessary slots.

defaults Object of class list, the relevant default values to be used when neither mapping nor args provides the necessary information.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)

[GdObject](#)

[GRanges](#)

[HighlightTrack](#)

[ImageMap](#)

[IRanges](#)

[RangeTrack](#)

[DataTrack](#)

[collapsing](#)

[grouping](#)

[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
# This is a reference class, below example from AlignmentsTrack

afrom <- 2960000
ato <- 3160000
alTrack <- AlignmentsTrack(system.file(
  package = "Gviz", "extdata",
  "gapped.bam"
), isPaired = TRUE)
plotTracks(alTrack, from = afrom, to = ato, chromosome = "chr12")
```

BiomartGeneRegionTrack-class

BiomartGeneRegionTrack class and methods

Description

A class to hold gene model data for a genomic region fetched dynamically from EBI's Biomart Ensembl data source.

Usage

```
## S4 method for signature 'BiomartGeneRegionTrack'
initialize(
  .Object,
  start = NULL,
  end = NULL,
  biomart,
  filter = list(),
  range,
  genome = NULL,
  chromosome = NULL,
  strand = NULL,
  featureMap = NULL,
  symbol = NULL,
  gene = NULL,
  transcript = NULL,
  entrez = NULL,
  ...
)

BiomartGeneRegionTrack(
```

```

    start = NULL,
    end = NULL,
    biomaRt,
    chromosome = NULL,
    strand,
    genome = NULL,
    stacking = "squish",
    filters = list(),
    featureMap = NULL,
    name = "BiomartGeneRegionTrack",
    symbol = NULL,
    gene = NULL,
    entrez = NULL,
    transcript = NULL,
    ...
)

## S4 method for signature 'BiomartGeneRegionTrack'
subset(x, from, to, chromosome, use.defaults = TRUE, ...)

```

Arguments

.Object	.Object
start	An integer scalar with the genomic start coordinates for the gene model range.
end	An integer scalar with the genomic end coordinates for the gene model range.
biomaRt	An optional Mart object providing access to the EBI Biomart webservice. As default the appropriate Ensembl data source is selected based on the provided genome and chromosome.
filter	filter
range	range
genome	The genome on which the track's ranges are defined. Usually this is a valid UCSC genome identifier, however this is not being formally checked at this point. If no mapping from genome to Biomart Ensembl data source is possible, the biomaRt argument needs to be provided by the user.
chromosome	The chromosome on which the track's genomic ranges are defined. A valid UCSC chromosome identifier. Please note that at this stage only syntactic checking takes place, i.e., the argument value needs to be a single integer, numeric character or a character of the form chr x , where x may be any possible string. The user has to make sure that the respective chromosome is indeed defined for the track's genome.
strand	Character scalar, the strand for which to fetch gene information from Biomart. One in +, -, or +/-.
featureMap	Named character vector or list to map between the fields in the Biomart data base and the features as they are used to construct the track. If multiple values are provided in a single list item, the package will use the first one that is defined in the selected Biomart.

symbol, transcript, gene, entrez	Character vector giving one or several gene symbols, Ensembl transcript identifiers, Ensembl gene identifiers, or ENTREZ gene identifiers, respectively. The genomic locus of their gene model will be fetch from Biomart instead of providing explicit start and end coordinates.
...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.
stacking	The stacking type for overlapping items of the track. One in c(hide, dense, squish, pack, full). Currently, only hide (don't show the track items, squish (make best use of the available space) and dense (no stacking at all) are implemented.
filters	A list of additional filters to be applied in the Biomart query. See biomaRt::getBM for details.
name	Character scalar of the track's name used in the title panel when plotting.
x	A valid track object class name, or the object itself, in which case the class is derived directly from it.
from, to	from, to
use.defaults	logical

Details

A track containing all gene models in a particular region as fetched from EBI's Biomart service. Usually the user does not have to take care of the Biomart connection, which will be established automatically based on the provided genome and chromosome information. However, for full flexibility a valid [Mart](#) object may be passed on to the constructor. Please note that this assumes a connection to one of the Ensembl gene data sources, mapping the available query data back to the internal object slots.

Value

The return value of the constructor function is a new object of class BiomartGeneRegionTrack.

Functions

- `initialize(BiomartGeneRegionTrack)`: Initialize display parameters and translate the various id/filter arguments (symbol, gene, transcript, entrez, filter) into the internal biomaRt query filters before deferring to the GeneRegionTrack initializer.
- `BiomartGeneRegionTrack()`: Constructor function for BiomartGeneRegionTrack-class.
- `subset(BiomartGeneRegionTrack)`: subset a BiomartGeneRegionTrack by coordinates and sort if necessary.

Objects from the class

Objects can be created using the constructor function BiomartGeneRegionTrack.

Author(s)

Florian Hahne

References

EBI Biomart webservice at <https://www.biomart.org>.

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## Construct the object
## Not run:
# create biomaRt object first
bm <- useEnsembl(
  biomaRt = "ENSEMBL_MART_ENSEMBL",
  dataset = "mmusculus_gene_ensembl"
)
# pass it to the `BiomartGeneRegionTrack`
bmTrack <- BiomartGeneRegionTrack(
  start = 26682683, end = 26711643,
  chromosome = 7, biomaRt = bm
)

## End(Not run)


## Plotting
plotTracks(bmTrack)


## Track names
names(bmTrack)
names(bmTrack) <- "foo"
plotTracks(bmTrack)
```

```
## Subsetting and splitting
subTrack <- subset(bmTrack, from = 26700000, to = 26705000)
length(subTrack)
subTrack <- bmTrack[transcript(bmTrack) == "ENSMUST00000144140"]
split(bmTrack, transcript(bmTrack))

## Accessors
start(bmTrack)
end(bmTrack)
width(bmTrack)
position(bmTrack)
width(subTrack) <- width(subTrack) + 100

strand(bmTrack)
strand(subTrack) <- "-"

chromosome(bmTrack)
chromosome(subTrack) <- "chrX"

genome(bmTrack)
genome(subTrack) <- "hg19"

range(bmTrack)
ranges(bmTrack)

## Annotation
identifier(bmTrack)
identifier(bmTrack, "lowest")
identifier(subTrack) <- "bar"

feature(bmTrack)
feature(subTrack) <- "foo"

exon(bmTrack)
exon(subTrack) <- letters[1:2]

gene(bmTrack)
gene(subTrack) <- "bar"

symbol(bmTrack)
symbol(subTrack) <- "foo"

transcript(bmTrack)
transcript(subTrack) <- c("foo", "bar")
chromosome(subTrack) <- "chr7"
plotTracks(subTrack)

values(bmTrack)

## Grouping
group(bmTrack)
group(subTrack) <- "Group 1"
transcript(subTrack)
```

```
plotTracks(subTrack)

## Stacking
stacking(bmTrack)
stacking(bmTrack) <- "dense"
plotTracks(bmTrack)

## coercion
as(bmTrack, "data.frame")
as(bmTrack, "UCSCData")

## HTML image map
coords(bmTrack)
tags(bmTrack)
bmTrack <- plotTracks(bmTrack)$foo
coords(bmTrack)
tags(bmTrack)
```

collapsing*Dynamic content based on the available resolution*

Description

When plotting features linearly along genomic coordinates one frequently runs into the problem of too little resolution to adequately display all details. Most genome browsers try to reasonably reduce the amount of detail that is shown based on the current zoom level.

Details

Most track classes in this package define an internal `collapseTrack` method which tries to adjust the plotted content to the available resolution, aims at reducing over-plotting and prevents rendering issues, e.g. when lines are too thin to be plotted. This feature can be toggled on or off using the `collapse` display parameter (see `settings` for details on setting these parameters).

In the simplest case (for `AnnotationTrack` objects) this involves expanding all shown features to a minimum pixel width and height (using display parameters `min.width` and `min.height`) and collapsing overlapping annotation items (as defined by the parameter `min.distance`) into one single item to prevent over-plotting.

For objects of class `DataTrack`, the data values underlying collapsed regions will be summarized based on the `summary` display parameter. See the class' documentation for more details.

See Also

[AnnotationTrack](#)

[DataTrack](#)

[settings](#)

CustomTrack-class *CustomTrack class and methods*

Description

A fully customizable track object to be populated via a user-defined plotting function.

Usage

```
## S4 method for signature 'CustomTrack'
initialize(.Object, plottingFunction, variables, ...)

CustomTrack(
  plottingFunction = function(GdObject, prepare = FALSE, ...) {
  },
  variables = list(),
  name = "CustomTrack",
  ...
)

## S4 method for signature 'CustomTrack'
drawGD(GdObject, minBase, maxBase, prepare = FALSE, ...)

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

Arguments

.Object	.Object
plottingFunction	A user-defined function to be executed once the track coordinates have been properly set up. The function needs to accept two mandatory arguments: GdObject, the CustomTrack object to be plotted, and prepare, a logical flag indicating whether the function has been called in preparation mode or in drawing mode. It also needs to return the input GdObject, potentially with modifications.
variables	A list of additional variables for the user-defined plotting function.
...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.
name	Character scalar of the track's name.
GdObject	Object of GdObject-class.
minBase	minBase
maxBase	maxBase
prepare	logical
object	object

Details

A track to allow for any sort of plotting, with the currently displayed genomic location set. Essentially this acts as a simple callback into the Gviz plotting machinery after all the track panels and coordinates have been set up. It is entirely up to the user what to plot in the track, or even to use the predefined coordinate system. The only prerequisite is that all plotting operations need to utilize Grid graphics.

Value

The return value of the constructor function is a new object of class CustomTrack.

Functions

- `initialize(CustomTrack)`: Initialize the `plottingFunction` and variables slots before deferring to the `GdObject` initializer for the remaining slots.
- `CustomTrack()`: Objects can be created using the constructor function.
- `drawGD(CustomTrack)`: plot the object to a graphics device. The return value of this method is the input object, potentially updated during the plotting operation. Internally, there are two modes in which the method can be called. Either in 'prepare' mode, in which case no plotting is done but the object is preprocessed based on the available space, or in 'plotting' mode, in which case the actual graphical output is created. Since subsetting of the object can be potentially costly, this can be switched off in case subsetting has already been performed before or is not necessary.
- `show(CustomTrack)`: Show method.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## Object construction:

## An empty object
CustomTrack()
```

datasets	<i>Data sets</i>
----------	------------------

Description

Some sample data sets used for the illustrative examples and the vignette.

DetailsAnnotationTrack-class
<i>The DetailsAnnotationTrack class directly extends AnnotationTrack</i>

Description

The purpose of this track type is to add an arbitrarily detailed plot section (typically consisting of additional quantitative data) for each range element of an AnnotationTrack. See the 'Details' section of the constructor documentation below for more information.

Usage

```
DetailsAnnotationTrack(...)

## S4 method for signature 'DetailsAnnotationTrack'
initialize(.Object, fun, selectFun, ...)

## S4 method for signature 'DetailsAnnotationTrack'
drawGD(GdObject, minBase, maxBase, prepare = FALSE, ...)
```

Arguments

...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.
.Object	.Object
fun	A function that is being called for each entry in the AnnotationTrack object. See section 'Details' and 'Examples' for further information. When called internally by the plotting machinery, a number of arguments are automatically passed on to this function, and the user needs to make sure that they can all be digested (i.e., either have all of them as formal named function arguments, or gobble up everything that is not needed in ...). These arguments are:

- start: the genomic start coordinate of the range item.
- end: the genomic end coordinates of the range item.
- strand: the strand information for the range item.
- chromosome: the chromosome of the range item.
- identifier: the identifier of the range item, i.e., the result of calling `identifier(DetailsAnnotationTrack, lowest=TRUE)`. Typically those identifiers are passed on to the object constructor during instantiation as the `id` argument.
- index: a counter enumerating the ranges. The `AnnotationTrack` object is sorted internally for visibility, and the `index` argument refers to the index of plotting.
- `GdObject`: a reference to the currently plotted `DetailsAnnotationTrack` object.
- `GdObject.original`: a reference to the `DetailsAnnotationTrack` before any processing like item collapsing has taken place. Essentially, this is the track object as it exists in your working environment. Additional arguments can be passed to the plotting function by means of the `detailsFunArgs` argument (see below). Note that the plot must use grid graphics (e.g. function in the 'lattice' package or low-level grid functions). To access a data object such a matrix or data frame within the function you can either store it as a variable in the global environment or, to avoid name space conflicts, you can make it part of the function environment by means of a closure. Alternatively, you may want to explicitly stick it into an environment or pass it along in the `detailsFunArgs` list. To figure out in your custom plotting function which annotation element is currently being plotted you can either use the identifier which has to be unique for each range element, or you may want to use the genomic position (start/end/strand/chromosome) e.g. if the data is stored in a `GRanges` object.

<code>selectFun</code>	A function that is being called for each entry in the <code>AnnotationTrack</code> object with exactly the same arguments as in <code>fun</code> . The purpose of this function is to decide for each track element whether details should be drawn, and consequently it has to return a single logical scalar. If the return value is <code>TRUE</code> , details will be drawn for the item, if it is <code>FALSE</code> , the details strip for the item is omitted.
<code>GdObject</code>	Object of <code>GdObject</code> -class
<code>minBase, maxBase</code>	<code>minBase, maxBase</code>
<code>prepare</code>	logical

Functions

- `DetailsAnnotationTrack()`: Constructor function for `DetailsAnnotationTrack`-class
The `DetailsAnnotationTrack` class directly extends `AnnotationTrack`. The purpose of this track type is to add an arbitrarily detailed plot section (typically consisting of additional quantitative data) for each range element of an `AnnotationTrack`. This allows a locus-wide view of annotation elements together with any kind of details per feature or element that may, for instance, provide insight on how some complex quantitative measurements change according

to their position in a locus. If the quantitative data is too complex for a DataTrack, e.g. because it requires extra space or a trellis-like representation, a DetailsAnnotationTrack can be used instead. Example: an AnnotationTrack shows the positions of a number of probes from a microarray, and you want a histogram of the signal intensity distribution derived from all samples at each of these probe locations. Another example usage would be to show, for each element of an AnnotationTrack, an xy-plot of the signal against some clinical measurement such as blood pressure. The limitation for applications of this type of track is basically only the available space of the device you are plotting to.

This flexibility is possible by utilizing a simple function model to perform all the detailed plotting. The functionality of this plotting function fun is totally up to the user, and the function environment is prepared in a way that all necessary information about the plotted annotation feature is available. To restrict the details section to only selected number of annotation features one can supply another function selectFun, which decides for each feature separately whether details are available or not. Finally, an arbitrary number of additional arguments can be passed on to these two function by means of the detailsFunArgs display parameter. This is expected to be a named list, and all list elements are passed along to the plotting function fun and to the selector function selectFun as additional named arguments. Please note that some argument names like start, end or identifier are reserved and can not be used in the detailsFunArgs list. For examples of plotting functions, see the 'Examples' section.

- `initialize(DetailsAnnotationTrack)`: Initialize the fun and selectFun slots before deferring to the AnnotationTrack initializer for the remaining slots.
- `drawGD(DetailsAnnotationTrack)`: plot the object to a graphics device. The return value of this method is the input object, potentially updated during the plotting operation. Internally, there are two modes in which the method can be called. Either in 'prepare' mode, in which case no plotting is done but the object is preprocessed based on the available space, or in 'plotting' mode, in which case the actual graphical output is created. Since subsetting of the object can be potentially costly, this can be switched off in case subsetting has already been performed before or is not necessary.

Slots

`fun` A function that is being called for each AnnotationTrack element to plot details.

`selectFun` A function that is being called for each AnnotationTrack element to decide whether details need to be plotted.

DisplayPars-class

DisplayPars: A class to control the plotting parameters for GdObjects

Description

All tracks within this package are highly customizable. The DisplayPars class facilitates this and provides a unified API to the customization parameters.

The individual parameters in a DisplayParameters class are stored as pointers in an environment. This has the upshot of not having to copy the whole track object when changing parameters, and parameters can be updated without the need to explicitly reassign the track to a symbol (i.e., updating

of parameters happens in place). The downside is that upon copying of track objects, the parameter environment needs to be re-instantiated.

Objects can be created using the constructor function `DisplayPars`.

The default display parameters for a track object class can be queried using the `availableDisplayPars` function.

Usage

```
DisplayPars(...)

getPar(x, name, ...)

## S4 method for signature 'DisplayPars,character'
getPar(x, name, asIs = FALSE)

## S4 method for signature 'DisplayPars,missing'
getPar(x, hideInternal = TRUE)

displayPars(x, name, ...)

## S4 method for signature 'DisplayPars,missing'
displayPars(x, hideInternal = TRUE)

## S4 method for signature 'DisplayPars,character'
displayPars(x, name)

## S4 method for signature 'DisplayPars'
as.list(x)

setPar(x, value, ...)

## S4 method for signature 'DisplayPars,list'
setPar(x, value, interactive = TRUE)

## S4 method for signature 'DisplayPars,character'
setPar(x, name, value, interactive = TRUE)

displayPars(x, recursive = FALSE) <- value

## S4 replacement method for signature 'DisplayPars,list'
displayPars(x, recursive = FALSE) <- value

## S4 method for signature 'DisplayPars'
show(object)

## S4 method for signature 'InferredDisplayPars'
as.list(x)
```

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

```
availableDisplayPars(class)
```

Arguments

...	All named arguments are stored in the object's environment as individual parameters, regardless of their type.
x	object to set the displayPar value on
name	Name of the retrieved parameter.
asIs	logical
hideInternal	logical
value	named value to be set
interactive	logical
recursive	logical
object	object
class	Either character scalar or object. Supported classes are: GdObject, GenomeAxisTrack, RangeTrack, NumericTrack, DataTrack, IdeogramTrack, StackedTrack, AnnotationTrack, DetailsAnnotationTrack, GeneRegionTrack, BiomartGeneRegionTrack, AlignmentsTrack, SequenceTrack, SequenceBSgenomeTrack, SequenceDNASTringSetTrack, SequenceRNASTringSetTrack

Value

The return value of the constructor function is a new object of class DisplayPars.

availableDisplayPars returns a list of the default display parameters.

Functions

- DisplayPars(): Constructor function.
 - getPar(): Generics for getPar.
 - getPar(x = DisplayPars, name = character): Return the value(s) of the display parameter(s) named in name, resolving parameter name aliases. getPar is the underlying implementation; displayPars is a more descriptive alias for it.
 - getPar(x = DisplayPars, name = missing): Return all display parameters as a named list, optionally hiding internal (.__-prefixed) parameters.
 - displayPars(): Generics for displayPars.
 - displayPars(x = DisplayPars, name = missing): Returns all available display parameters.
 - displayPars(x = DisplayPars, name = character): Returns the value of a subset of display parameters, as identified by name.
 - as.list(DisplayPars): Converts DisplayPars to list.
 - setPar(): Generics for SetPar.
- SetPar generic function

- `setPar(x = DisplayPars, value = list)`: Sets display parameters by the values of the named list in value. Note that display parameters in the `DisplayPars`-class are pass-by-reference, so no re-assignment to the symbol obj is necessary.
- `setPar(x = DisplayPars, value = character)`: set the single display parameter name to value. Note that display parameters in the `DisplayPars` class are pass-by-reference, so no re-assignment to the symbol obj is necessary.
- `displayPars(x, recursive = FALSE) <- value`: Generics for `displayPars<-`.
- `displayPars(x = DisplayPars) <- value`: Replaces or adds display parameters as provided by the named list items.
- `show(DisplayPars)`: Show method.
- `as.list(InferredDisplayPars)`: Return `InferredDisplayPars` as a list.
- `show(InferredDisplayPars)`: Show method.
- `availableDisplayPars()`: Get default display parameters.

Slots

`pars` an environment or a list containing parameter key value pairs.

Author(s)

Florian Hahne

Examples

```
## Construct object
dp <- DisplayPars(col = "red", lwd = 2, transformation = log2)
dp

## Query parameters
displayPars(dp)
displayPars(dp, "col")
getPar(dp, c("col", "transformation"))

## Modify parameters
displayPars(dp) <- list(lty = 1, fontsize = 3)
setPar(dp, "pch", 20)
dp

## Default parameters
availableDisplayPars("GenomeAxisTrack")
```

exportTracks

Export Gviz tracks into an annotation file representation.

Description

This function is still a bit experimental. Write all tracks provided as a list `tracks` into a single BED file. So far only BED export is supported.

Usage

```
exportTracks(tracks, range, chromosome, file)
```

Arguments

tracks	A list of annotation track objects to be exported into a single BED file.
range	A numeric vector or length 2. The genomic range to display when opening the file in a browser.
chromosome	The chromosome to display when opening the file in a browser.
file	Character, the path to the file to write into.

Value

The function is called for its side effect of writing to a file.

Author(s)

Florian Hahne

Examples

```
## export AnnotationTrack to BED file
at <- AnnotationTrack(start = seq(1, 10), width = 1, chromosome = "chr1")
exportTracks(list(at),
  range = c(1, 10), chromosome = "chr1",
  file = paste0(tempfile(), ".bed")
)
```

GeneRegionTrack-class *GeneRegionTrack class and methods*

Description

A class to hold gene model data for a genomic region.

Usage

```
## S4 method for signature 'GeneRegionTrack'
initialize(.Object, start, end, ...)

## S4 method for signature 'ReferenceGeneRegionTrack'
initialize(
  .Object,
  stream,
  reference,
  mapping = list(),
  args = list(),
```

```
    defaults = list(),
    ...
)

GeneRegionTrack(
  range = NULL,
  rstarts = NULL,
  rends = NULL,
  rwidths = NULL,
  strand,
  feature,
  exon,
  transcript,
  gene,
  symbol,
  chromosome,
  genome,
  stacking = "squish",
  name = "GeneRegionTrack",
  start = NULL,
  end = NULL,
  importFunction,
  stream = FALSE,
  ...
)

## S4 method for signature 'GeneRegionTrack'
gene(GdObject)

## S4 replacement method for signature 'GeneRegionTrack,character'
gene(GdObject) <- value

## S4 method for signature 'GeneRegionTrack'
symbol(GdObject)

## S4 replacement method for signature 'GeneRegionTrack,character'
symbol(GdObject) <- value

## S4 method for signature 'GeneRegionTrack'
transcript(GdObject)

## S4 replacement method for signature 'GeneRegionTrack,character'
transcript(GdObject) <- value

## S4 method for signature 'GeneRegionTrack'
exon(GdObject)

## S4 replacement method for signature 'GeneRegionTrack,character'
```

```

exon(GdObject) <- value

## S4 method for signature 'GeneRegionTrack'
group(object)

## S4 replacement method for signature 'GeneRegionTrack,character'
group(object) <- value

## S4 method for signature 'GeneRegionTrack'
identifier(
  GdObject,
  type = .dpOrDefault(GdObject, "transcriptAnnotation", "symbol")
)

## S4 replacement method for signature 'GeneRegionTrack,character'
identifier(GdObject) <- value

## S4 method for signature 'ReferenceGeneRegionTrack'
subset(x, ...)

## S4 method for signature 'GeneRegionTrack'
drawGD(GdObject, ...)

## S4 method for signature 'GeneRegionTrack'
show(object)

## S4 method for signature 'ReferenceGeneRegionTrack'
show(object)

```

Arguments

.Object	.Object
start,end	An integer scalar with the genomic start or end coordinate for the gene model range. If those are missing, the default value will automatically be the smallest (or largest) value, respectively in <code>rstarts</code> and <code>rends</code> for the currently active chromosome. When building a <code>GeneRegionTrack</code> from a <code>TxDb</code> object, these arguments can be used to subset the desired annotation data by genomic coordinates. Please note this in that case the chromosome parameter must also be set.
...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.
stream	A logical flag indicating that the user-provided import function can deal with indexed files and knows how to process the additional selection argument when accessing the data on disk. This causes the constructor to return a <code>ReferenceGeneRegionTrack</code> object which will grab the necessary data on the fly during each plotting operation.
reference	reference file

mapping	mapping
args	args
defaults	logical
range	

An optional meta argument to handle the different input types. If the range argument is missing, all the relevant information to create the object has to be provided as individual function arguments (see below).

The different input options for range are:

A TxDb object: all the necessary gene model information including exon locations, transcript groupings and associated gene ids are contained in TxDb objects, and the coercion between the two is almost completely automated. If desired, the data to be fetched from the TxDb object can be restricted using the constructor's chromosome, start and end arguments. See below for details. A direct coercion method `as(obj, "GeneRegionTrack")` is also available. A nice added benefit of this input option is that the UTR and coding region information that is part of the original TxDb object is retained in the GeneRegionTrack.

A GRanges object: the genomic ranges for the GeneRegion track as well as the optional additional metadata columns feature, transcript, gene, exon and symbol (see description of the individual function parameters below for details). Calling the constructor on a GRanges object without further arguments, e.g. `GeneRegionTrack(range=obj)` is equivalent to calling the coerce method `as(obj, "GeneRegionTrack")`.

A GRangesList object: this is very similar to the previous case, except that the grouping information that is part of the list structure is preserved in the GeneRegionTrack. I.e., all the elements within one list item receive the same group id. For consistency, there is also a coercion method from GRangesLists `as(obj, "GeneRegionTrack")`. Please note that unless the necessary information about gene ids, symbols, etc. is present in the individual GRanges meta data slots, the object will not be particularly useful, because all the identifiers will be set to a common default value.

An IRanges object: almost identical to the GRanges case, except that the chromosome and strand information as well as all additional data has to be provided in the separate chromosome, strand, feature, transcript, symbol, exon or gene arguments, because it can not be directly encoded in an IRanges object. Note that only the former two are mandatory (if not provided explicitly the more or less reasonable default values `chromosome=NA` and `strand=*` are used, but not providing information about the gene-to-transcript relationship or the human-readable symbols renders a lot of the class' functionality useless.

A data.frame object: the data.frame needs to contain at least the two mandatory columns `start` and `end` with the range coordinates. It may also contain a `chromosome` and a `strand` column with the chromosome and strand information for each range. If missing, this information will be drawn from the constructor's `chromosome` or `strand` arguments. In addition, the `feature`, `exon`, `transcript`, `gene` and `symbol` data can be provided as columns in the data.frame. The above comments about potential default values also apply here.

	A character scalar: in this case the value of the range argument is considered to be a file path to an annotation file on disk. A range of file types are supported by the Gviz package as identified by the file extension. See the <code>importFunction</code> documentation below for further details.
<code>rstarts</code>	An integer vector of the start coordinates for the actual gene model items, i.e., for the individual exons. The relationship between exons is handled via the gene and transcript factors. Alternatively, this can be a vector of comma-separated lists of integer coordinates, one vector item for each transcript, and each comma-separated element being the start location of a single exon within that transcript. Those lists will be exploded upon object instantiation and all other annotation arguments will be recycled accordingly to regenerate the exon/transcript/gene relationship structure. This implies the appropriate number of items in all annotation and coordinates arguments.
<code>rends</code>	An integer vector of the end coordinates for the actual gene model items. Both <code>rstarts</code> and <code>rends</code> have to be of equal length.
<code>rwidths</code>	An integer vector of widths for the actual gene model items. This can be used instead of either <code>rstarts</code> or <code>rends</code> to specify the range coordinates.
<code>strand</code>	Character vector, the strand information for the individual track exons. It may be provided in the form <code>+</code> for the Watson strand, <code>-</code> for the Crick strand or <code>*</code> for either one of the two. Please note that all items within a single gene or transcript model need to be on the same strand, and erroneous entries will result in casting of an error.
<code>feature</code>	Factor (or other vector that can be coerced into one), giving the feature types for the individual track exons. When plotting the track to the device, if a display parameter with the same name as the value of <code>feature</code> is set, this will be used as the track item's fill color. Additionally, the feature type defines whether an element in the <code>GeneRegionTrack</code> is considered to be coding or non-coding. The details section as well as the section about the <code>thinBoxFeature</code> display parameter further below has more information on this. See also grouping for details.
<code>exon</code>	Character vector of exon identifiers. It's values will be used as the identifier tag when plotting to the device if the display parameter <code>showExonId=TRUE</code>.
<code>transcript</code>	Factor (or other vector that can be coerced into one), giving the transcript memberships for the individual track exons. All items with the same transcript identifier will be visually connected when plotting to the device. See grouping for details. Will be used as labels when <code>showId=TRUE</code>, and <code>geneSymbol=FALSE</code>.
<code>gene</code>	Factor (or other vector that can be coerced into one), giving the gene memberships for the individual track exons.
<code>symbol</code>	A factor with human-readable gene name aliases which will be used as labels when <code>showId=TRUE</code>, and <code>geneSymbol=TRUE</code>.
<code>chromosome</code>	The chromosome on which the track's genomic ranges are defined. A valid UCSC chromosome identifier if <code>options(ucscChromosomeNames=TRUE)</code>. Please note that in this case only syntactic checking takes place, i.e., the argument value needs to be an integer, numeric character or a character of the form <code>chrX</code>, where <code>X</code> may be any possible string. The user has to make sure that the respective

	chromosome is indeed defined for the the track's genome. If not provided here, the constructor will try to build the chromosome information based on the available inputs, and as a last resort will fall back to the value chrNA. Please note that by definition all objects in the Gviz package can only have a single active chromosome at a time (although internally the information for more than one chromosome may be present), and the user has to call the chromosome<- replacement method in order to change to a different active chromosome. When creating a GeneRegionTrack from a TxDb object, the value of this parameter can be used to subset the data to fetch only transcripts from a single chromosome.
genome	The genome on which the track's ranges are defined. Usually this is a valid UCSC genome identifier, however this is not being formally checked at this point. If not provided here the constructor will try to extract this information from the provided inputs, and eventually will fall back to the default value of NA.
stacking	The stacking type for overlapping items of the track. One in c(hide, dense, squish, pack, full). Currently, only hide (don't show the track items, squish (make best use of the available space) and dense (no stacking at all) are implemented.
name	Character scalar of the track's name used in the title panel when plotting.
importFunction	A user-defined function to be used to import the data from a file. This only applies when the range argument is a character string with the path to the input data file. The function needs to accept an argument x containing the file path and has to return a proper GRanges object with all the necessary metadata columns set. A set of default import functions is already implemented in the package for a number of different file types, and one of these defaults will be picked automatically based on the extension of the input file name. If the extension can not be mapped to any of the existing import function, an error is raised asking for a user-defined import function via this argument. Currently the following file types can be imported with the default functions: gff, gff1, gff2, gff3, gtf.
GdObject	Object of GdObject-class.
value	Value to be set.
object	object
type	type
x	A valid track object class name, or the object itself, in which case the class is derived directly from it.

Details

A track containing all gene models in a particular region. The data are usually fetched dynamically from an online data store, but it is also possible to manually construct objects from local data. Connections to particular online data sources should be implemented as sub-classes, and GeneRegionTrack is just the common denominator that is being used for plotting later on. There are several levels of data associated to a GeneRegionTrack:

exon level: identifiers are stored in the exon column of the [GRanges](#) object in the range slot. Data may be extracted using the exon method.

transcript level: identifiers are stored in the transcript column of the [GRanges](#) object. Data may be extracted using the transcript method.

gene level: identifiers are stored in the gene column of the [GRanges](#) object, more human-readable versions in the symbol column. Data may be extracted using the gene or the symbol methods.

transcript-type level: information is stored in the feature column of the [GRanges](#) object. If a display parameter of the same name is specified, the software will use its value for the coloring.

GeneRegionTrack objects also know about coding regions and non-coding regions (e.g., UTRs) in a transcript, and will indicate those by using different shapes (wide boxes for all coding regions, thinner boxes for non-coding regions). This is archived by setting the feature values of the object for non-coding elements to one of the options that are provided in the thinBoxFeature display parameters. All other elements are considered to be coding elements.

Value

The return value of the constructor function is a new object of class GeneRegionTrack.

Functions

- `initialize(GeneRegionTrack)`: Initialize the start and end slots before deferring to the AnnotationTrack initializer for the remaining slots.
- `initialize(ReferenceGeneRegionTrack)`: Initialize the ReferenceTrack slots (stream, reference, mapping, args, defaults) before deferring to the GeneRegionTrack initializer for the remaining slots.
- `GeneRegionTrack()`: Constructor function for GeneRegionTrack-class.
- `gene(GeneRegionTrack)`: Extract the gene identifiers for all gene models.
- `gene(GdObject = GeneRegionTrack) <- value`: Replace the gene identifiers for all gene models. The replacement value must be a character of appropriate length or another vector that can be coerced into such.
- `symbol(GeneRegionTrack)`: Extract the human-readable gene symbol for all gene models.
- `symbol(GdObject = GeneRegionTrack) <- value`: Replace the human-readable gene symbol for all gene models. The replacement value must be a character of appropriate length or another vector that can be coerced into such.
- `transcript(GeneRegionTrack)`: Extract the transcript identifiers for all transcripts in the gene models.
- `transcript(GdObject = GeneRegionTrack) <- value`: Replace the transcript identifiers for all transcripts in the gene model. The replacement value must be a character of appropriate length or another vector that can be coerced into such.
- `exon(GeneRegionTrack)`: Extract the exon identifiers for all exons in the gene models.
- `exon(GdObject = GeneRegionTrack) <- value`: replace the exon identifiers for all exons in the gene model. The replacement value must be a character of appropriate length or another vector that can be coerced into such.
- `group(GeneRegionTrack)`: extract the group membership for all track items.
- `group(object = GeneRegionTrack) <- value`: replace the grouping information for track items. The replacement value must be a factor of appropriate length or another vector that can be coerced into such.

- `identifier(GeneRegionTrack)`: return track item identifiers. Depending on the setting of the optional argument `lowest`, these are either the group identifiers or the individual item identifiers. export
- `identifier(GdObject = GeneRegionTrack) <- value`: Set the track item identifiers. The replacement value has to be a character vector of appropriate length. This always replaces the group-level identifiers, so essentially it is similar to `groups<-`.
- `subset(ReferenceGeneRegionTrack)`: Subset a `GeneRegionTrack` by coordinates and sort if necessary.
- `drawGD(GeneRegionTrack)`: plot the object to a graphics device. The return value of this method is the input object, potentially updated during the plotting operation. Internally, there are two modes in which the method can be called. Either in 'prepare' mode, in which case no plotting is done but the object is preprocessed based on the available space, or in 'plotting' mode, in which case the actual graphical output is created. Since subsetting of the object can be potentially costly, this can be switched off in case subsetting has already been performed before or is not necessary.
- `show(GeneRegionTrack)`: Show method.
- `show(ReferenceGeneRegionTrack)`: Show method.

Objects from the class

Objects can be created using the constructor function `GeneRegionTrack`.

Author(s)

Florian Hahne, Steve Lianoglou

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```

## The empty object
GeneRegionTrack()

## Load some sample data
data(cyp2b10)

## Construct the object
grTrack <- GeneRegionTrack(
  start = 26682683, end = 26711643,
  rstart = cyp2b10$start, rends = cyp2b10$end, chromosome = 7, genome = "mm39",
  transcript = cyp2b10$transcript, gene = cyp2b10$gene, symbol = cyp2b10$symbol,
  feature = cyp2b10$feature, exon = cyp2b10$exon,
  name = "Cyp2b10", strand = cyp2b10$strand
)

## Directly from the data.frame
grTrack <- GeneRegionTrack(cyp2b10)

## From a TxDb object
if (require(GenomicFeatures)) {
  samplefile <- system.file("extdata",
                             "hg19_knownGene_sample.sqlite",
                             package = "GenomicFeatures")
  txdb <- loadDb(samplefile)
  GeneRegionTrack(txdb)
  GeneRegionTrack(txdb, chromosome = "chr6", start = 35000000, end = 40000000)
}

## Plotting
plotTracks(grTrack)

## Track names
names(grTrack)
names(grTrack) <- "foo"
plotTracks(grTrack)

## Subsetting and splitting
subTrack <- subset(grTrack, from = 25597000, to = 25610000)
length(subTrack)
subTrack <- grTrack[transcript(grTrack) == "ENSMUST00000005477"]
split(grTrack, transcript(grTrack))

## Accessors
start(grTrack)
end(grTrack)
width(grTrack)
position(grTrack)
width(subTrack) <- width(subTrack) + 100

```

```
strand(grTrack)
strand(subTrack) <- "-"

chromosome(grTrack)
chromosome(subTrack) <- "chrX"

genome(grTrack)
genome(subTrack) <- "mm39"

range(grTrack)
ranges(grTrack)

## Annotation
identifier(grTrack)
identifier(grTrack, "lowest")
identifier(subTrack) <- "bar"

feature(grTrack)
feature(subTrack) <- "foo"

exon(grTrack)
exon(subTrack) <- rep(letters[1:2], 6)

gene(grTrack)
gene(subTrack) <- "bar"

symbol(grTrack)
symbol(subTrack) <- "foo"

transcript(grTrack)
transcript(subTrack) <- "foo"
chromosome(subTrack) <- "chr7"
plotTracks(subTrack)

values(grTrack)

## Grouping
group(grTrack)
group(subTrack) <- "Group 1"
transcript(subTrack)
plotTracks(subTrack)

## Collapsing transcripts
plotTracks(grTrack,
  collapseTranscripts = TRUE, showId = TRUE,
  extend.left = 10000, shape = "arrow"
)

## Stacking
stacking(grTrack)
stacking(grTrack) <- "dense"
plotTracks(grTrack)
```

```
## coercion
as(grTrack, "data.frame")
as(grTrack, "UCSCData")

## HTML image map
coords(grTrack)
tags(grTrack)
grTrack <- plotTracks(grTrack)$foo
coords(grTrack)
tags(grTrack)
```

grouping

Grouping of annotation features

Description

Many annotation tracks are actually composed of a number of grouped sub-features, for instance exons in a gene model. This man page highlights the use of grouping information to build informative annotation plots.

Details

All track objects that inherit from class [AnnotationTrack](#) support the grouping feature. The information is usually passed on to the constructor function (for [AnnotationTrack](#) via the `group` argument and for [GeneRegionTrack](#) objects via the `exon` argument) or automatically downloaded from an online annotation repository ([BiomartGeneRegionTrack](#)). Group membership is specified by a factor vector with as many items as there are annotation items in the track (i.e., the value of `length(track)`). Upon plotting, the grouped annotation features are displayed together and will not be separated in the stacking of track items.

Author(s)

Florian Hahne

See Also

[AnnotationTrack](#)

[BiomartGeneRegionTrack](#)

[GeneRegionTrack](#)

Gviz-defunct	<i>Defunct functions in package Gviz</i>
--------------	--

Description

These functions are defunct and no longer available.

Defunct functions are::

(none)

Gviz-deprecated	<i>Deprecated functions in package Gviz</i>
-----------------	---

Description

These functions are provided for compatibility with older versions of Gviz only, and will be defunct at the next release.

The following functions are deprecated and will be made defunct (use the replacement indicated below)::

(none)

HighlightTrack-class	<i>HighlightTrack class and methods</i>
----------------------	---

Description

A container for other track objects from the Gviz package that allows for the addition of a common highlighting area across tracks.

Usage

```
## S4 method for signature 'HighlightTrack'
initialize(Object, trackList, ...)
```

```
HighlightTrack(
  trackList = list(),
  range = NULL,
  start = NULL,
  end = NULL,
  width = NULL,
  chromosome,
  genome,
  name = "HighlightTrack",
```

```

    ...
)

## S4 replacement method for signature 'HighlightTrack,list'
displayPars(x, recursive = FALSE) <- value

## S4 method for signature 'HighlightTrack'
length(x)

## S4 replacement method for signature 'HighlightTrack'
chromosome(GdObject) <- value

## S4 method for signature 'HighlightTrack'
setStacks(GdObject, ...)

## S4 method for signature 'HighlightTrack'
consolidateTrack(GdObject, chromosome, ...)

## S4 method for signature 'HighlightTrack'
subset(x, ...)

## S4 method for signature 'HighlightTrack'
show(object)

```

Arguments

<code>.Object</code>	<code>.Object</code>
<code>trackList</code>	A list of Gviz track objects that all have to inherit from class <code>GdObject</code> .
<code>...</code>	All additional parameters are ignored.
<code>range</code>	An optional meta argument to handle the different input types. If the range argument is missing, all the relevant information to create the object has to be provided as individual function arguments (see below). The different input options for range are: A GRanges object: the genomic ranges for the highlighting regions. An IRanges object: almost identical to the GRanges case, except that the chromosome information has to be provided in the separate chromosome argument, because it can not be directly encoded in an IRanges object. A data.frame object: the data.frame needs to contain at least the two mandatory columns <code>start</code> and <code>end</code> with the range coordinates. It may also contain a chromosome column with the chromosome information for each range. If missing, this information will be drawn from the constructor's chromosome argument.
<code>start, end</code>	An integer scalar with the genomic start or end coordinates for the highlighting range. Can also be supplied as part of the range argument.
<code>width</code>	An integer vector of widths for highlighting ranges. This can be used instead of either <code>start</code> or <code>end</code> to specify the range coordinates.

chromosome	The chromosome on which the track's genomic ranges are defined. A valid UCSC chromosome identifier if <code>options(ucscChromosomeNames=TRUE)</code> . Please note that in this case only syntactic checking takes place, i.e., the argument value needs to be an integer, numeric character or a character of the form <code>chrX</code> , where <code>x</code> may be any possible string. The user has to make sure that the respective chromosome is indeed defined for the track's genome. If not provided here, the constructor will try to build the chromosome information based on the available inputs, and as a last resort will fall back to the value <code>chrNA</code> . Please note that by definition all objects in the <code>Gviz</code> package can only have a single active chromosome at a time (although internally the information for more than one chromosome may be present), and the user has to call the <code>chromosome<-replacement</code> method in order to change to a different active chromosome.
genome	The genome on which the track's ranges are defined. Usually this is a valid UCSC genome identifier, however this is not being formally checked at this point. If not provided here the constructor will try to extract this information from the provided inputs, and eventually will fall back to the default value of <code>NA</code> .
name	Character scalar of the track's name. This is not really used and only exists for completeness.
x	A valid track object class name, or the object itself, in which case the class is derived directly from it.
recursive	logical
value	Value to be set.
GdObject	Object of <code>GdObject</code> -class.
object	object

Details

A track to conceptually group other `Gviz` track objects into a meta track for the sole purpose of overlaying all the contained tracks with the same highlighting region as defined by the objects genomic ranges. During rendering the contained tracks will be treated as if they had been provided to the `plotTracks` function as individual objects.

Value

The return value of the constructor function is a new object of class `HighlightTrack`.

Functions

- `initialize(HighlightTrack)`: Initialize the `trackList` slot before deferring to the `RangeTrack` initializer for the remaining slots.
- `HighlightTrack()`: Constructor function for `HighlightTrack`-class.
- `displayPars(x = HighlightTrack) <- value`: set display parameters using the values of the named list in `value`. See [settings](#) for details on display parameters and customization.
- `length(HighlightTrack)`: return the number of subtracks.

- `chromosome(HighlightTrack) <- value`: replace the value of the track's chromosome. This has to be a valid UCSC chromosome identifier or an integer or character scalar that can be reasonably coerced into one.
- `setStacks(HighlightTrack)`: Recompute the stacks based on the available space and on the object's track items and stacking settings. This really just calls the `setStacks` methods for the contained tracks and only exists for dispatching reasons.
- `consolidateTrack(HighlightTrack)`: For a `HighlightTrack`, apply the `consolidateTrack` method on each of the subtracks in the `trackList` slot.
- `subset(HighlightTrack)`: subset all the contained tracks in a `HighlightTrack` by coordinates and sort if necessary.
- `show(HighlightTrack)`: Show method.

Objects from the Class

Objects can be created using the constructor function `HighlightTrack`.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## Object construction:
set.seed(123)
dat <- runif(100, min = -2, max = 22)
gt <- GenomeAxisTrack()
dt <- DataTrack(data = dat, start = sort(sample(200, 100)), width = 1, genome = "hg19")

ht <- HighlightTrack(trackList = list(gt, dt))
```

IdeogramTrack-class IdeogramTrack class and methods

Description

A class to represent the schematic display of a chromosome, also known as an ideogram. The respective information is typically directly fetched from UCSC.

Usage

```
## S4 method for signature 'IdeogramTrack'
initialize(.Object, genome, chromosome, bands, name, ...)

IdeogramTrack(chromosome = NULL, genome, name = NULL, bands = NULL, ...)

## S4 method for signature 'IdeogramTrack'
start(x)

## S4 replacement method for signature 'IdeogramTrack'
start(x) <- value

## S4 method for signature 'IdeogramTrack'
end(x)

## S4 replacement method for signature 'IdeogramTrack'
end(x) <- value

## S4 method for signature 'IdeogramTrack'
width(x)

## S4 replacement method for signature 'IdeogramTrack'
width(x) <- value

## S4 method for signature 'IdeogramTrack'
length(x)

## S4 replacement method for signature 'IdeogramTrack'
chromosome(GdObject) <- value

## S4 replacement method for signature 'IdeogramTrack'
genome(x) <- value

## S4 method for signature 'IdeogramTrack,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'IdeogramTrack'
position(GdObject, ...)
```

```
## S4 method for signature 'IdeogramTrack'
drawGD(GdObject, minBase, maxBase, prepare = FALSE, ...)
```

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

Arguments

.Object	The object being initialized (standard S4 initialize convention); not normally supplied directly by the user.
genome	The genome on which to create the ideogram. This has to be a valid UCSC genome identifier if the ideogram data is to be fetched from the UCSC repository.
chromosome	The chromosome for which to create the ideogram. Has to be a valid UCSC chromosome identifier of the form chr x , or a single integer or numeric character unless option(ucscChromosomeNames=FALSE). The user has to make sure that the respective chromosome is indeed defined for the the track's genome.
bands	A data.frame with the cytoband information for all available chromosomes on the genome similar to the data that would be fetched from UCSC. The table needs to contain the mandatory columns chrom, chromStart, chromEnd, name and gieStain with the chromosome name, cytoband start and end coordinates, cytoband name and coloring information, respectively. This can be used when no connection to the internet is available or when the cytoband information has been cached locally to avoid the somewhat slow connection to UCSC.
name	Character scalar of the track's name used in the title panel when plotting. Defaults to the selected chromosome.
...	Additional items which will all be interpreted as further display parameters.
x, GdObject, object	The IdeogramTrack object.
value	The replacement value for a replacement method.
i, j	Index arguments for [-style subsetting; ignored, since subsetting is not supported for IdeogramTrack.
drop	logical; ignored, since subsetting is not supported for IdeogramTrack.
minBase, maxBase	The currently plotted genomic range.
prepare	logical, indicating whether drawGD is being called in 'prepare' mode (compute layout only) or 'plotting' mode (render to the device).

Details

Ideograms are schematic depictions of chromosomes, including chromosome band information and centromere location. The relevant data for various species is stored in the UCSC data base. The initializer method of the class will automatically fetch the respective data for a given genome and chromosome from UCSC and fill the appropriate object slots. When plotting IdeogramTrack objects, the current genomic location is indicated on the chromosome by a colored box.

The `Gviz.ucscUrl` option controls which URL is being used to connect to UCSC. For instance, one could switch to the European UCSC mirror by calling `options(Gviz.ucscUrl = "http://genome-euro.ucsc.edu/cgi-bin/ucsc/ideogram/ideogram.pl")`.

Value

The return value of the constructor function is a new object of class `IdeogramTrack`.

Functions

- `initialize(IdeogramTrack)`: Fetch the chromosome band and length information from UCSC (or use the user-supplied bands table) and use it to populate the range and bandTable slots.
- `IdeogramTrack()`: Constructor function for `IdeogramTrack`-class.
- `start(IdeogramTrack)`: returns `NULL`. The start coordinate is not a meaningful concept for a whole-chromosome `IdeogramTrack` and is not supported.
- `start(IdeogramTrack) <- value`: a no-op: the start coordinate cannot be set for an `IdeogramTrack` and the object is returned unchanged.
- `end(IdeogramTrack)`: returns `NULL`. The end coordinate is not a meaningful concept for a whole-chromosome `IdeogramTrack` and is not supported.
- `end(IdeogramTrack) <- value`: a no-op: the end coordinate cannot be set for an `IdeogramTrack` and the object is returned unchanged.
- `width(IdeogramTrack)`: returns `NULL`. The width is not a meaningful concept for a whole-chromosome `IdeogramTrack` and is not supported.
- `width(IdeogramTrack) <- value`: a no-op: the width cannot be set for an `IdeogramTrack` and the object is returned unchanged.
- `length(IdeogramTrack)`: return the number of chromosome bands stored in the object.
- `chromosome(IdeogramTrack) <- value`: replace the active chromosome. If band information for the new chromosome is already cached in the object's `bandTable` slot it is reused directly; otherwise a new `IdeogramTrack` is constructed by re-fetching the band data for the new chromosome.
- `genome(IdeogramTrack) <- value`: replace the active genome, re-fetching the chromosome band information for the new genome.
- `x = IdeogramTrack[i = ANY, j = ANY, drop = ANY]`: subsetting is not supported for `IdeogramTrack`; the object is returned unchanged.
- `position(IdeogramTrack)`: returns `NULL`. Genomic position is not a meaningful concept for a whole-chromosome `IdeogramTrack` and is not supported.
- `drawGD(IdeogramTrack)`: plot the object to a graphics device, rendering the chromosome bands, centromere and, if `from/to` are supplied, a highlighted box indicating the currently displayed genomic region.
- `show(IdeogramTrack)`: Show method.

Objects from the Class

Objects can be created using the constructor function `IdeogramTrack`.

Note

When fetching ideogram data from UCSC the results are cached for faster access. See [clearSessionCache](#) on details to delete these cached items.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## Construct the object
## Not run:
idTrack <- IdeogramTrack(chromosome = 7, genome = "mm39")

## End(Not run)


## Plotting
plotTracks(idTrack, from = 5000000, to = 9000000)


## Track names
names(idTrack)
names(idTrack) <- "foo"
plotTracks(idTrack, from = 5000000, to = 9000000)


## Accessors
chromosome(idTrack)
```

```
## Not run:
chromosome(idTrack) <- "chrX"

## End(Not run)

genome(idTrack)
## Not run:
genome(id) <- "hg38"

## End(Not run)

range(idTrack)
ranges(idTrack)

## Annotation
values(idTrack)

## coercion
as(idTrack, "data.frame")
```

NumericTrack-class	<i>NumericTrack class and methods</i>
--------------------	---------------------------------------

Description

The virtual parent class for all track items in the Gviz package designed to contain numeric data. This class merely exists for dispatching purpose.

Usage

```
## S4 method for signature 'NumericTrack'
drawAxis(GdObject, from, to, ...)

## S4 method for signature 'NumericTrack'
drawGrid(GdObject, from, to)
```

Arguments

GdObject	Object of GdObject-class.
from, to	Numeric scalar, giving the range of genomic coordinates to limit the tracks in. Note that to cannot be larger than from.
...	Additional arguments.

Value

A virtual class: No objects may be created from it.

Functions

- `drawAxis(NumericTrack)`: add a y-axis to the title panel of a track.
- `drawGrid(NumericTrack)`: superpose a grid on top of a track.

Slots

- `dp` Object of `DisplayPars`-class, the display settings controlling the look and feel of a track. See settings for details on setting graphical parameters for tracks.
- `name` Object of class `character`, a human-readable name for the track that will be used in the track's annotation panel if necessary.
- `imageMap` Object of `ImageMap`-class, containing optional information for an HTML image map. This will be created by the `drawGD` methods when the track is plotted to a device and is usually not set by the user.
- `range` Object of class `GRanges`, the genomic ranges of the track items as well as additional annotation information in its `elementMetadata` slot. Please note that the slot is actually implemented as a class union between `GRanges` and `IRanges` to increase efficiency, for instance for `DataTrack` objects. This usually does not concern the user.
- `chromosome` Object of class `character`, the chromosome on which the track is defined. There can only be a single chromosome for one track. For certain subclasses, the space of allowed chromosome names is limited (e.g., only those chromosomes that exist for a particular genome). Throughout the package, chromosome names have to be entered either as a single integer scalar or as a character scalar of the form `chrXYZ`, where `XYZ` may be an arbitrary character string.
- `genome` Object of class `character`, the genome for which the track is defined. For most subclasses this has to be a valid UCSC genome identifier, however this may not always be formally checked upon object instantiation.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)

[plotTracks](#)
[settings](#)

plotTracks	<i>The main plotting function for one or several Gviz tracks.</i>
------------	---

Description

plotTracks is the main interface when plotting single track objects, or lists of tracks linked together across the same genomic coordinates. Essentially, the resulting plots are very similar to the graphical output of the UCSC Genome Browser, except for all of the interactivity.

Usage

```
plotTracks(  
  trackList,  
  from = NULL,  
  to = NULL,  
  ...,  
  sizes = NULL,  
  panel.only = FALSE,  
  extend.right = 0,  
  extend.left = 0,  
  title.width = NULL,  
  add = FALSE,  
  main,  
  cex.main = 2,  
  fontface.main = 2,  
  col.main = "black",  
  margin = 6,  
  chromosome = NULL,  
  innerMargin = 3  
)
```

Arguments

trackList	A list of Gviz track objects, all inheriting from class GdObject . The tracks will all be drawn to the same genomic coordinates, either as defined by the from and to arguments if supplied, or by the maximum range across all individual items in the list.
from, to	Numeric scalar, giving the range of genomic coordinates to draw the tracks in. Note that to cannot be larger than from. If NULL, the plotting ranges are derived from the individual tracks. See extend.left and extend.right below for the definition of the final plotting ranges.

...	Additional arguments which are all interpreted as display parameters to tweak the appearance of the plot. These parameters are global, meaning that they will be used for all tracks in the list where they actually make sense, and they override the track-internal settings. See settings for details on display parameters.
sizes	A numeric vector of relative vertical sizes for the individual tracks of length equal to the number of tracks in trackList, or NULL to auto-detect the most appropriate vertical size proportions.
panel.only	Logical flag, causing the tracks to be plotted as lattice-like panel functions without resetting the plotting canvas and omitting the title pane. This allows to embed tracks into a trellis layout. Usually the function is called for a single track only when panel.only==TRUE.
extend.right, extend.left	Numeric scalar, extend the plotting range to the right or to the left by a fixed number of bases. The final plotting range is defined as from-extend.left to to+extend.right.
title.width	An expansion factor for the width of the title panels. This can be used to make more space, e.g. to accommodate for more detailed data axes. The default is to use as much space as needed to fit all the annotation text.
add	Logical flag, add the plot to an existing plotting canvas without re-initialising.
main	Character scalar, the plot's main header.
cex.main, fontface.main, col.main	The fontface, color and expansion factor settings for the main header.
margin	The margin width to add to the plot in pixels.
chromosome	Set the chromosome for all the tracks in the track list.
innerMargin	The inner margin width to add to the plot in pixels.

Details

Gviz tracks are plotted in a vertically stacked layout. Each track panel is split up into a title section containing the track name, as well as an optional axis for tracks containing numeric data, and a data section showing the actual data along genomic coordinates. In that sense, the output is very similar to the UCSC Genome Browser.

The layout of the individual tracks is highly customizable though so called "display parameters". See [settings](#) for details.

While plotting a track, the software automatically computes HTML image map coordinates based on the current graphics device. These coordinates as well as the associated annotation information can later be used to embed images of the plots in semi-interactive HTML pages. See [ImageMap](#) for details.

Value

A list of Gviz tracks, each one augmented by the computed image map coordinates in the imageMap slot, along with the additional ImageMap object titles containing information about the title panels.

Author(s)

Florian Hahne

See Also[GdObject](#)[ImageMap](#)[StackedTrack](#)[settings](#)**Examples**

```
## Create some tracks to plot
st <- c(2000000, 2070000, 2100000, 2160000)
ed <- c(2050000, 2130000, 2150000, 2170000)
str <- c("-", "+", "-", "-")
gr <- c("Group1", "Group2", "Group1", "Group3")
annTrack <- AnnotationTrack(
  start = st, end = ed, strand = str, chromosome = 7,
  genome = "hg19", feature = "test", group = gr,
  id = paste("annTrack item", 1:4),
  name = "annotation track foo",
  stacking = "squish"
)

ax <- GenomeAxisTrack()

dt <- DataTrack(
  start = seq(min(st), max(ed), len = 10), width = 18000,
  data = matrix(runif(40), nrow = 4), genome = "hg19", chromosome = 7,
  type = "histogram", name = "data track bar"
)

## Now plot the tracks
res <- plotTracks(list(ax, annTrack, dt))

## Plot only a subrange
res <- plotTracks(list(ax, annTrack, dt), from = 2080000, to = 2156000)

## Extend plotting ranges
res <- plotTracks(list(ax, annTrack, dt), extend.left = 200000, extend.right = 200000)

## Add a header
res <- plotTracks(list(ax, annTrack, dt),
  main = "A GenomGraphs plot",
  col.main = "darkgray"
)

## Change vertical size and title width
```

```

res <- plotTracks(list(ax, annTrack, dt), sizes = c(1, 1, 5))

names(annTrack) <- "foo"
res <- plotTracks(list(ax, annTrack), title.width = 0.6)

## Adding and lattice like plots
library(grid)
grid.newpage()
pushViewport(viewport(height = 0.5, y = 1, just = "top"))
grid.rect()
plotTracks(annTrack, add = TRUE)
popViewport(1)
pushViewport(viewport(height = 0.5, y = 0, just = "bottom"))
grid.rect()
plotTracks(dt, add = TRUE)
popViewport(1)
## Not run:
library(lattice)
myPanel <- function(x, ...) {
  plotTracks(annTrack,
    panel.only = TRUE,
    from = min(x), to = max(x), shape = "box"
  )
}
a <- seq(19000000, 22500000, len = 40)
xyplot(b ~ a | c, data.frame(a = a, b = 1, c = cut(a, 4)),
  panel = myPanel,
  scales = list(x = "free")
)

## End(Not run)

```

RangeTrack-class

RangeTrack class and methods

Description

The virtual parent class for all track items in the Gviz package that contain some form of genomic ranges (start, end, strand, chromosome and the associated genome.)

Usage

```

## S4 method for signature 'RangeTrack'
initialize(.Object, range, chromosome, genome, ...)

## S4 method for signature 'RangeTrack'
ranges(x)

## S4 replacement method for signature 'RangeTrack'

```

```
ranges(x) <- value

## S4 method for signature 'RangeTrack'
range(x)

## S4 method for signature 'RangeTrack'
seqnames(x)

## S4 method for signature 'RangeTrack'
seqlevels(x)

## S4 method for signature 'RangeTrack'
seqinfo(x)

## S4 method for signature 'RangeTrack'
genome(x)

## S4 replacement method for signature 'RangeTrack'
genome(x) <- value

## S4 method for signature 'RangeTrack'
chromosome(GdObject)

## S4 replacement method for signature 'RangeTrack'
chromosome(GdObject) <- value

## S4 method for signature 'RangeTrack'
start(x)

## S4 replacement method for signature 'RangeTrack'
start(x) <- value

## S4 method for signature 'RangeTrack'
end(x)

## S4 replacement method for signature 'RangeTrack'
end(x) <- value

## S4 method for signature 'RangeTrack'
width(x)

## S4 replacement method for signature 'RangeTrack'
width(x) <- value

## S4 method for signature 'RangeTrack'
min(x)

## S4 method for signature 'RangeTrack'
```

```

max(x)

## S4 method for signature 'RangeTrack'
length(x)

## S4 method for signature 'RangeTrack'
strand(x)

## S4 replacement method for signature 'RangeTrack,ANY'
strand(x) <- value

## S4 method for signature 'RangeTrack'
position(GdObject, from = NULL, to = NULL, sort = FALSE, ...)

## S4 method for signature 'RangeTrack,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'RangeTrack'
subset(
  x,
  from = NULL,
  to = NULL,
  sort = FALSE,
  drop = TRUE,
  use.defaults = TRUE,
  ...
)

## S4 method for signature 'RangeTrack,ANY'
split(x, f, drop = FALSE, ...)

## S4 method for signature 'RangeTrack'
values(x)

## S4 method for signature 'RangeTrack'
feature(GdObject)

## S4 replacement method for signature 'RangeTrack,character'
feature(GdObject) <- value

## S4 method for signature 'RangeTrack'
consolidateTrack(GdObject, chromosome, ...)

```

Arguments

.Object	.Object
range	range
chromosome	the currently active chromosome which may have to be set for a RangeTrack or

	a SequenceTrack object
genome	genome
...	Additional arguments.
x	A valid track object class name, or the object itself, in which case the class is derived directly from it.
value	Value to be set.
GdObject	the input track object
from, to	Numeric scalar, giving the range of genomic coordinates to limit the tracks in. Note that to cannot be larger than from.
sort	logical.
i	Numeric scalar, index to subset.
j	Numeric scalar, index to subset. Ignored.
drop	logical, indicating if levels that do not occur should be dropped (if f is a factor).
use.defaults	logical.
f	factor in the sense that <code>as.factor(f)</code> defines the grouping,

Value

A virtual class: No objects may be created from it.

Functions

- `initialize(RangeTrack)`: Initialize the range, chromosome and genome slots before deferring to the GdObject initializer for the remaining slots.
- `ranges(RangeTrack)`: return the genomic coordinates for the track along with all additional annotation information as an object of class GRanges.
- `ranges(RangeTrack) <- value`: replace the genomic coordinates and associated annotation information for the track with a new GRanges object.
- `range(RangeTrack)`: return the genomic coordinates for the track as an object of class IRanges.
- `seqnames(RangeTrack)`: return the track's seqnames.
- `seqlevels(RangeTrack)`: return the track's seqlevels.
- `seqinfo(RangeTrack)`: return the track's seqinfo.
- `genome(RangeTrack)`: return the track's genome.
- `genome(RangeTrack) <- value`: set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.
- `chromosome(RangeTrack)`: return the chromosome for which the track is defined.
- `chromosome(RangeTrack) <- value`: replace the value of the track's chromosome. This has to be a valid UCSC chromosome identifier or an integer or character scalar that can be reasonably coerced into one.
- `start(RangeTrack)`: the start of the track items in genomic coordinates.

- `start(RangeTrack) <- value`: replace the start of the track items in genomic coordinates.
- `end(RangeTrack)`: the end of the track items in genomic coordinates.
- `end(RangeTrack) <- value`: replace the end of the track items in genomic coordinates.
- `width(RangeTrack)`: the width of the track items in genomic coordinates.
- `width(RangeTrack) <- value`: replace the width of the track items in genomic coordinates.
- `min(RangeTrack)`: return the start position for the leftmost range item.
- `max(RangeTrack)`: return the end position for the rightmost range item.
- `length(RangeTrack)`: return the number of items in the track.
- `strand(RangeTrack)`: return a vector of strand specifiers for all track items, in the form '+' for the Watson strand, '-' for the Crick strand or '*' for either of the two.
- `strand(x = RangeTrack) <- value`: replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.
- `position(RangeTrack)`: the arithmetic mean of the track item's coordinates, i.e., $(\text{end}(\text{obj}) - \text{start}(\text{obj}))/2$.
- `x = RangeTrack[i = ANY, j = ANY, drop = ANY]`: subset the items in the RangeTrack object. This is essentially similar to subsetting of the GRanges object in the range slot. For most applications, the subset method may be more appropriate.
- `subset(RangeTrack)`: subset a RangeTrack by coordinates and sort if necessary.
- `split(x = RangeTrack, f = ANY)`: split a RangeTrack object by an appropriate factor vector (or another vector that can be coerced into one). The output of this operation is a list of objects of the same class as the input object, all inheriting from class RangeTrack.
- `values(RangeTrack)`: return all additional annotation information except for the genomic coordinates for the track items as a data.frame.
- `feature(RangeTrack)`: return the grouping information for track items. For certain subclasses, groups may be indicated by different colour schemes when plotting. See grouping or AnnotationTrack and GeneRegionTrack for details. @export
- `feature(GdObject = RangeTrack) <- value`: set the grouping information for track items. This has to be a factor vector (or another type of vector that can be coerced into one) of the same length as the number of items in the RangeTrack. See grouping or AnnotationTrack and GeneRegionTrack for details. @export
- `consolidateTrack(RangeTrack)`: Update the active chromosome (if provided) before deferring to the GdObject method for the remaining consolidation steps.

Slots

- dp** Object of DisplayPars-class, the display settings controlling the look and feel of a track. See settings for details on setting graphical parameters for tracks.
- name** Object of class character, a human-readable name for the track that will be used in the track's annotation panel if necessary.
- imageMap** Object of ImageMap-class, containing optional information for an HTML image map. This will be created by the drawGD methods when the track is plotted to a device and is usually not set by the user.

range Object of class GRanges, the genomic ranges of the track items as well as additional annotation information in its elementMetadata slot. Please note that the slot is actually implemented as a class union between GRanges and IRanges to increase efficiency, for instance for DataTrack objects. This usually does not concern the user.

chromosome Object of class character, the chromosome on which the track is defined. There can only be a single chromosome for one track. For certain subclasses, the space of allowed chromosome names is limited (e.g., only those chromosomes that exist for a particular genome). Throughout the package, chromosome names have to be entered either as a single integer scalar or as a character scalar of the form chrXYZ, where XYZ may be an arbitrary character string.

genome Object of class character, the genome for which the track is defined. For most sub-classes this has to be a valid UCSC genome identifier, however this may not always be formally checked upon object instantiation.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## This is a reference class therefore we show below
## an example from AnnotationTrack

## An empty object
AnnotationTrack()

## Construct from individual arguments
st <- c(2000000, 2070000, 2100000, 2160000)
ed <- c(2050000, 2130000, 2150000, 2170000)
str <- c("-", "+", "-", "-")
gr <- c("Group1", "Group2", "Group1", "Group3")
```

```

annTrack <- AnnotationTrack(
  start = st, end = ed, strand = str, chromosome = 7,
  genome = "hg19", feature = "test", group = gr,
  id = paste("annTrack item", 1:4),
  name = "generic annotation", stacking = "squish"
)

## Plotting
plotTracks(annTrack)

```

SequenceBSgenomeTrack-class

The BSgenome-based version of the SequenceTrack class

Description

The BSgenome-based version of the SequenceTrack class

Usage

```

## S4 method for signature 'SequenceBSgenomeTrack'
initialize(.Object, sequence = NULL, ...)

```

Arguments

.Object	.Object
sequence	sequence
...	Additional arguments.

Functions

- `initialize(SequenceBSgenomeTrack)`: Initialize the sequence and pointerCache slots before deferring to the SequenceTrack initializer for the remaining slots.

Slots

sequence A BSgenome object (or NULL), realized on demand as individual chromosome sequences are accessed.

pointerCache An environment holding pointers to the BSgenome sequences that have already been realized, to prevent garbage collection.

SequenceDNAStringSetTrack-class

The DNAStringSet-based version of the SequenceTrack class

Description

The DNAStringSet-based version of the SequenceTrack class

Usage

```
## S4 method for signature 'SequenceDNAStringSetTrack'
initialize(.Object, sequence, ...)
```

Arguments

.Object	.Object
sequence	sequence
...	Additional arguments.

Functions

- initialize(SequenceDNAStringSetTrack): Initialize the sequence slot (defaulting to an empty DNAStringSet if none is supplied) before deferring to the SequenceTrack initializer for the remaining slots.

Slots

sequence A DNAStringSet object containing the sequence data, with one element per chromosome.

SequenceRNAStringSetTrack-class

The RNAStringSet-based version of the SequenceTrack class

Description

The RNAStringSet-based version of the SequenceTrack class

Usage

```
## S4 method for signature 'SequenceRNAStringSetTrack'
initialize(.Object, sequence, ...)
```

Arguments

.Object	.Object
sequence	sequence
...	Additional arguments.

Functions

- `initialize(SequenceRNAStringSetTrack)`: Initialize the sequence slot (defaulting to an empty `RNAStringSet` if none is supplied) before deferring to the `SequenceTrack` initializer for the remaining slots.

Slots

`sequence` A `RNAStringSet` object containing the sequence data, with one element per chromosome.

SequenceTrack-class *SequenceTrack class and methods*

Description

A track class to represent genomic sequences. The three child classes `SequenceDNAStringSetTrack`, `SequenceRNAStringSetTrack` and `SequenceBSgenomeTrack` do most of the work, however in practise they are of no particular relevance to the user.

Usage

```
## S4 method for signature 'SequenceTrack'
initialize(.Object, chromosome, genome, ...)
```

```
SequenceTrack(
  sequence,
  chromosome,
  genome,
  name = "SequenceTrack",
  importFunction,
  stream = FALSE,
  ...
)
```

```
RNASequenceTrack(
  sequence,
  chromosome,
  genome,
  name = "SequenceTrack",
  importFunction,
```

```
    stream = FALSE,
    ...
)

## S4 method for signature 'SequenceTrack'
seqnames(x)

## S4 method for signature 'SequenceBSgenomeTrack'
seqnames(x)

## S4 method for signature 'SequenceTrack'
seqlevels(x)

## S4 method for signature 'SequenceBSgenomeTrack'
seqlevels(x)

## S4 method for signature 'SequenceTrack'
start(x)

## S4 method for signature 'SequenceTrack'
end(x)

## S4 method for signature 'SequenceTrack'
width(x)

## S4 method for signature 'SequenceTrack'
length(x)

## S4 method for signature 'SequenceTrack'
subseq(x, start = NA, end = NA, width = NA)

## S4 method for signature 'ReferenceSequenceTrack'
subseq(x, start = NA, end = NA, width = NA)

## S4 method for signature 'SequenceTrack'
chromosome(GdObject)

## S4 replacement method for signature 'SequenceTrack'
chromosome(GdObject) <- value

## S4 method for signature 'SequenceTrack'
genome(x)

## S4 method for signature 'SequenceTrack'
consolidateTrack(GdObject, chromosome, ...)

## S4 method for signature 'SequenceTrack'
drawGD(GdObject, minBase, maxBase, prepare = FALSE, ...)
```

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

## S4 method for signature 'SequenceDNASTringSetTrack'
show(object)

## S4 method for signature 'SequenceRNASTringSetTrack'
show(object)

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

Arguments

.Object	.Object
chromosome	the currently active chromosome which may have to be set for a RangeTrack or a SequenceTrack object parameters
genome	The genome on which the track's ranges are defined. Usually this is a valid UCSC genome identifier, however this is not being formally checked at this point. For a SequenceBSgenomeTrack object, the genome information is extracted from the input BSgenome package. For a DNASTringSet it has too be provided or the constructor will fall back to the default value of NA.
...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.
sequence	A meta argument to handle the different input types, making the construction of a SequenceTrack as flexible as possible. The different input options for sequence are: • An object of class DNASTringSet: the individual DNASTrings are considered to be the different chromosome sequences. • An object of class BSgenome: the Gviz package tries to follow the BSgenome philosophy in that the respective chromosome sequences are only realized once they are first accessed. • A character scalar: the value of the sequence argument is considered to be a file path to an annotation file on disk. A range of file types are supported by the Gviz package as identified by the file extension. See the importFunction documentation below for further details.
name	Character scalar of the track's name used in the title panel when plotting.
importFunction	A user-defined function to be used to import the sequence data from a file. This only applies when the sequence argument is a character string with the path to the input data file. The function needs to accept an argument file containing the file path and has to return a proper DNASTringSet object with the sequence information per chromosome. A set of default import functions is already implemented in the package for a number of different file types, and one of these defaults will be picked automatically based on the extension of the input file name. If the extension can not be mapped to any of the existing import function, an error is raised asking for a user-defined import function. Currently the

following file types can be imported with the default functions: *fa/fasta* and *2bit*. Both file types support indexing by genomic coordinates, and it makes sense to only load the part of the file that is needed for plotting. To this end, the *Gviz* package defines the derived *ReferenceSequenceTrack* class, which supports streaming data from the file system. The user typically does not have to deal with this distinction but may rely on the constructor function to make the right choice as long as the default import functions are used. However, once a user-defined import function has been provided and if this function adds support for indexed files, you will have to make the constructor aware of this fact by setting the *stream* argument to *TRUE*. Please note that in this case the import function needs to accept a second mandatory argument *selection* which is a *GRanges* object containing the dimensions of the plotted genomic range. As before, the function has to return an appropriate *DNAStringSet* object.

<i>stream</i>	A logical flag indicating that the user-provided import function can deal with indexed files and knows how to process the additional selection argument when accessing the data on disk. This causes the constructor to return a <i>ReferenceSequenceTrack</i> object which will grab the necessary data on the fly during each plotting operation.
<i>x</i>	A valid track object class name, or the object itself, in which case the class is derived directly from it.
<i>start, end, width</i>	Integer scalars defining the requested subsequence. Exactly two of the three must be provided; the third is derived from them.
<i>GdObject</i>	the input track object
<i>value</i>	Value to be set.
<i>minBase</i>	Start of the sequence.
<i>maxBase</i>	End of the sequence.
<i>prepare</i>	logical
<i>object</i>	object

Value

The return value of the constructor function is a new object of class *SequenceDNAStringSetTrack*, *SequenceBSgenomeTrack* or *ReferenceSequenceTrack*, depending on the constructor arguments. Typically the user will not have to be troubled with this distinction and can rely on the constructor to make the right choice.

Functions

- *initialize(SequenceTrack)*: Initialize the display parameter defaults and the chromosome/genome slots before deferring to the *GdObject* initializer for the remaining slots.
- *SequenceTrack()*: Constructor
- *RNASequenceTrack()*: Constructor
- *seqnames(SequenceTrack)*: return the names (i.e., the chromosome) of the sequences contained in the object.

- `seqnames(SequenceBSgenomeTrack)`: return the names (i.e., the chromosome) of the sequences contained in the object.
- `seqlevels(SequenceTrack)`: return the names (i.e., the chromosome) of the sequences contained in the object. Only those with `length > 0`.
- `seqlevels(SequenceBSgenomeTrack)`: return the names (i.e., the chromosome) of the sequences contained in the object. Only those with `length > 0`.
- `start(SequenceTrack)`: return the start coordinates of the track items.
- `end(SequenceTrack)`: return the end coordinates of the track items.
- `width(SequenceTrack)`: return the width of the track items in genomic coordinates.
- `length(SequenceTrack)`: return the length of the sequence for active chromosome.
- `subseq(SequenceTrack)`: extract a subsequence for the active chromosome between `start` and `end` (or of the given `width`), padding with `-` outside the available sequence range. Two of `start`, `end` and `width` must be provided.
- `subseq(ReferenceSequenceTrack)`: extract a subsequence for the active chromosome by streaming the required region from the referenced file, rather than from an in-memory sequence.
- `chromosome(SequenceTrack)`: return the chromosome for which the track is defined.
- `chromosome(SequenceTrack) <- value`: replace the value of the track's chromosome. This has to be a valid UCSC chromosome identifier or an integer or character scalar that can be reasonably coerced into one.
- `genome(SequenceTrack)`: Set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.
- `consolidateTrack(SequenceTrack)`: Consolidate/ Determine whether there is chromosome settings or not, and add this information.
- `drawGD(SequenceTrack)`: plot the object to a graphics device. The return value of this method is the input object, potentially updated during the plotting operation. Internally, there are two modes in which the method can be called. Either in 'prepare' mode, in which case no plotting is done but the object is preprocessed based on the available space, or in 'plotting' mode, in which case the actual graphical output is created. Since subsetting of the object can be potentially costly, this can be switched off in case subsetting has already been performed before or is not necessary.
- `show(SequenceBSgenomeTrack)`: Show method.
- `show(SequenceDNASTringSetTrack)`: Show method.
- `show(SequenceRNASTringSetTrack)`: Show method.
- `show(ReferenceSequenceTrack)`: Show method.

Objects from the class

Objects can be created using the constructor function `SequenceTrack`.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)
[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## An empty object
SequenceTrack()

## Construct from DNASTringSet
library(Biostrings)
letters <- c("A", "C", "T", "G", "N")
set.seed(999)
seqs <- DNASTringSet(c(chr1 = paste(sample(letters, 100000, TRUE),
  collapse = ""
), chr2 = paste(sample(letters, 200000, TRUE), collapse = "")))
sTrack <- SequenceTrack(seqs, genome = "hg38")
sTrack

## Construct from BSgenome object
if (require(BSgenome.Hsapiens.UCSC.hg38)) {
  sTrack <- SequenceTrack(Hsapiens)
  sTrack
}

## Set active chromosome
chromosome(sTrack)
chromosome(sTrack) <- "chr2"
head(seqnames(sTrack))

## Plotting
## Sequences
plotTracks(sTrack, from = 199970, to = 200000)
## Boxes
```

```

plotTracks(sTrack, from = 199800, to = 200000)
## Line
plotTracks(sTrack, from = 1, to = 200000)
## Force boxes
plotTracks(sTrack, from = 199970, to = 200000, noLetters = TRUE)
## Direction indicator
plotTracks(sTrack, from = 199970, to = 200000, add53 = TRUE)
## Sequence complement
plotTracks(sTrack, from = 199970, to = 200000, add53 = TRUE, complement = TRUE)
## Colors
plotTracks(sTrack, from = 199970, to = 200000, add53 = TRUE, fontcolor = c(
  A = 1,
  C = 1, G = 1, T = 1, N = 1
))

## Track names
names(sTrack)
names(sTrack) <- "foo"

## Accessors
genome(sTrack)
genome(sTrack) <- "mm39"
length(sTrack)

## Sequence extraction
subseq(sTrack, start = 100000, width = 20)
## beyond the stored sequence range
subseq(sTrack, start = length(sTrack), width = 20)

```

settings

Setting display parameters to control the look and feel of the plots

Description

The genome track plots in this package are all highly customizable by means of so called 'display parameters'. This page highlights the use of these parameters and list all available settings for the different track classes.

Arguments

scheme	A named nested list of display parameters, where the first level of nesting represents Gviz track object classes, and the second level of nesting represents parameters.
name	A character scalar with the scheme name.

Details

All of the package's track objects inherit the `dp` slot from the [GdObject](#) parent class, which is the main container to store an object's display parameters. Internally, the content of this slot has to be an

object of class `DisplayPars`, but the user is usually not exposed to this low level implementation. Instead, there are two main interaction points, namely the individual object constructor functions and the final `plotTracks` function. In both cases, all additional arguments that are not caught by any of the formally defined function parameters are being interpreted as additional display parameters and are automatically added to the aforementioned slot. The main difference here is that display parameters that are passed on to the constructor function are specific for an individual track object, whereas those supplied to the `plotTracks` function will be applied to all the objects in the plotting list. Not all display parameters have an effect on the plotting of all track classes, and those will be silently ignored.

One can query the available display parameters for a given class as well as their default values by calling the `availableDisplayPars` function, or by inspecting the man pages of the individual track classes. The structure of the classes defined in this package is hierarchical, and so are the available display parameters, i.e., all objects inherit the parameters defined in the common `GdObject` parent class, and so on.

Once a track object has been created, the display parameters are still open for modification. To this end, the `DisplayPars` replacement method is available for all objects inheriting from class `GdObject`. The method takes a named list of parameters as input, e.g.:

```
displayPars(foo) <- list(col="red", lwd=2)
```

In the same spirit, the currently set display parameters for the object `foo` can be inferred using the `displayPars` method directly, e.g.:

```
displayPars(foo)
```

For track objects inheriting from class `AnnotationTrack`, display parameters that are not formally defined in the class definition or in any of the parent classes are considered to be valid R color identifiers that are used to distinguish between different types of annotation features. For instance, the parameter `'miRNA'` will be used to color all annotation features of class `miRNA`. The annotation types can be set in the constructor function of the track object via the `feature` argument. For most of the tracks that have been inferred from one of the online repositories, this classification will usually be downloaded along with the actual annotation data.

Users might find themselves changing the same parameters over and over again, and it would make sense to register these modifications in a central location once and for all. To this end the `Gviz` package supports display parameter schemes. A scheme is essentially just a bunch of nested named lists, where the names on the first level of nesting should correspond to track class names, and the names on the second level to the display parameters to set. The currently active scheme can be changed by setting the global option `Gviz.scheme`, and a new scheme can be registered by using the `addScheme` function, providing both the list and the name for the new scheme. The `getScheme` function is useful to get the current scheme as a list structure, for instance to use as a skeleton for your own custom scheme.

In order to make these settings persistent across R sessions one can create one or several schemes in the global environment in the special object `.GvizSchemes`, for instance by putting the necessary code in the `.Rprofile` file. This object needs to be a named list of schemes, and it will be collected when the `Gviz` package loads. Its content is then automatically added to the collection of available schemes.

Please note that because display parameters are stored with the track objects, a scheme change only has an effect on those objects that are created after the change has taken place.

Value

The `getScheme` function returns current scheme as a list structure.

Display Parameters

GenomeAxisTrack: `add35=FALSE`: Logical scalar. Add 3' to 5' direction indicators.
`add53=FALSE`: Logical scalar. Add 5' to 3' direction indicators.
`background.title="transparent"`: Character scalar. The background color for the title panel. Defaults to omit the background.
`cex.id=0.7`: Numeric scalar. The text size for the optional range annotation.
`cex=0.8`: Numeric scalar. The overall font expansion factor for the axis annotation text.
`col.border.title="transparent"`: Integer or character scalar. The border color for the title panels.
`lwd.border.title=1`: Integer scalar. The border width for the title panels.
`col.id="white"`: Character scalar. The text color for the optional range annotation.
`col.range="cornsilk4"`: Character scalar. The border color for highlighted regions on the axis.
`distFromAxis=1`: Numeric scalar. Control the distance of the axis annotation from the tick marks.
`exponent=NULL`: Numeric scalar. The exponent for the axis coordinates, e.g., 3 means mb, 6 means gb, etc. The default is to automatically determine the optimal exponent.
`fill.range="cornsilk3"`: Character scalar. The fill color for highlighted regions on the axis.
`fontcolor="#808080"`: Character scalar. The font color for the axis annotation text.
`fontsize=10`: Numeric scalar. Font size for the axis annotation text in points.
`labelPos="alternating"`: Character vector, one in "alternating", "revAlternating", "above" or "below". The vertical positioning of the axis labels. If `scale` is not NULL, the possible values are "above", "below" and "beside".
`littleTicks=FALSE`: Logical scalar. Add more fine-grained tick marks.
`lwd=2`: Numeric scalar. The line width for the axis elementes.
`scale=NULL`: Numeric scalar. If not NULL a small scale is drawn instead of the full axis, if the value is between 0 and 1 it is interpreted as a fraction of the current plotting region, otherwise as an absolute length value in genomic coordinates.
`showId=FALSE`: Logical scalar. Show the optional range highlighting annotation.
`showTitle=FALSE`: Logical scalar. Plot a title panel. Defaults to omit the title panel.
`ticksAt=NULL`: Numeric scalar. The exact x-position for tickmarks (in base-pairs).
`size=NULL`: Numeric scalar. The relative size of the track. Can be overridden in the `plotTracks` function. Defaults to the ideal size based on the other track settings.
`col="darkgray"`: Character scalar. The color for the axis lines and tickmarks.

Inherited from class GdObject:

`alpha=1`: Numeric scalar. The transparency for all track items.
`alpha.title=NULL`: Numeric scalar. The transparency for the title panel.
`background.panel="transparent"`: Integer or character scalar. The background color of the content panel.

`background.legend="transparent"`: Integer or character scalar. The background color for the legend.

`cex.axis=NULL`: Numeric scalar. The expansion factor for the axis annotation. Defaults to `NULL`, in which case it is automatically determined based on the available space.

`cex.title=NULL`: Numeric scalar. The expansion factor for the title panel. This effects the fontsize of both the title and the axis, if any. Defaults to `NULL`, which means that the text size is automatically adjusted to the available space.

`col.axis="white"`: Integer or character scalar. The font and line color for the y axis, if any.

`col.frame="lightgray"`: Integer or character scalar. The line color used for the panel frame, if `frame==TRUE`

`col.grid="#808080"`: Integer or character scalar. Default line color for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`col.line=NULL`: Integer or character scalar. Default colors for plot lines. Usually the same as the global `col` parameter.

`col.symbol=NULL`: Integer or character scalar. Default colors for plot symbols. Usually the same as the global `col` parameter.

`col.title="white"`: Integer or character scalar. The border color for the title panels

`collapse=TRUE`: Boolean controlling whether to collapse the content of the track to accommodate the minimum current device resolution. See [collapsing](#) for details.

`fill="lightgray"`: Integer or character scalar. Default fill color setting for all plotting elements, unless there is a more specific control defined elsewhere.

`fontface.title=2`: Integer or character scalar. The font face for the title panels.

`fontface=1`: Integer or character scalar. The font face for all text, unless a more specific definition exists.

`fontfamily.title="sans"`: Integer or character scalar. The font family for the title panels.

`fontfamily="sans"`: Integer or character scalar. The font family for all text, unless a more specific definition exists.

`frame=FALSE`: Boolean. Draw a frame around the track when plotting.

`grid=FALSE`: Boolean, switching on/off the plotting of a grid.

`h=-1`: Integer scalar. Parameter controlling the number of horizontal grid lines, see [lattice::panel.grid](#) for details.

`lineheight=1`: Numeric scalar. The font line height for all text, unless a more specific definition exists.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`lty="solid"`: Numeric scalar. Default line type setting for all plotting elements, unless there is a more specific control defined elsewhere.

`lwd.title=1`: Integer scalar. The border width for the title panels

`lwd.grid=1`: Numeric scalar. Default line width for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`min.distance=1`: Numeric scalar. The minimum pixel distance before collapsing range items, only if `collapse==TRUE`. See [collapsing](#) for details.

`min.height=3`: Numeric scalar. The minimum range height in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`min.width=1`: Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`reverseStrand=FALSE`: Logical scalar. Set up the plotting coordinates in 3' -> 5' direction if TRUE. This will effectively mirror the plot on the vertical axis.

`rotation.title=90`: The rotation angle for the text in the title panel. Even though this can be adjusted, the automatic resizing of the title panel will currently not work, so use at own risk.

`rotation=0`: The rotation angle for all text unless a more specific definition exists.

`showAxis=TRUE`: Boolean controlling whether to plot a y axis (only applies to track types where axes are implemented).

`v=-1`: Integer scalar. Parameter controlling the number of vertical grid lines, see [lattice::panel.grid](#) for details.

DataTrack: `aggregateGroups=FALSE`: Logical scalar. Aggregate the values within a sample group using the aggregation function specified in the aggregation parameter.

`aggregation="mean"`: Function or character scalar. Used to aggregate values in windows or for collapsing overlapping items. The function has to accept a numeric vector as a single input parameter and has to return a numeric scalar with the aggregated value. Alternatively, one of the predefined options mean, median, sum, min, max or extreme can be supplied as a character scalar. Defaults to mean.

`missingAsZero=TRUE`: Logical scalar. Defines how the missing values are treated in the aggregation procedure with running window. Setting it to TRUE fills empty positions with zeros, which is default. FALSE fills empty positions with NA.

`alpha.confint=0.3`: Numeric scalar. The transparency for the confidence intervals in confint-type plots.

`amount=NULL`: Numeric scalar. Amount of jittering in xy-type plots. See [lattice::panel.xyplot](#) for details.

`baseline=NULL`: Numeric scalar. Y-axis position of an optional baseline. This parameter has a special meaning for mountain-type and polygon-type plots, see the 'Details' section in [DataTrack](#) for more information.

`box.legend=FALSE`: Logical scalar. Draw a box around a legend.

`box.ratio=1`: Numeric scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

`box.width=NULL`: Numeric scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

`grid=FALSE`: Logical vector. Draw a line grid under the track content.

`cex.legend=0.8`: Numeric scalar. The size factor for the legend text.

`cex.sampleNames=NULL`: Numeric scalar. The size factor for the sample names text in heatmap or horizon plots. Defaults to an automatic setting.

`cex=0.7`: Numeric scalar. The default pixel size for plotting symbols.

`coef=1.5`: Numeric scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

`col.baseline=NULL`: Character scalar. Color for the optional baseline, defaults to the setting of `col`.

`col.confint=NA`: Character vector. Border colors for the confidence intervals for confint-type plots.

`col.bboxplotFrame="#808080"`: Character scalar. Line color of the frame around grouped boxplots.

`col.histogram="#808080"`: Character scalar. Line color in histogram-type plots.

`col.horizon=NA`: The line color for the segments in the horizon-type plot. See [latticeExtra::panel.horizonplot](#) for details.

`col.mountain=NULL`: Character scalar. Line color in mountain-type and polygon-type plots, defaults to the setting of `col`.

`col.sampleNames="white"`: Character or integer scalar. The color used for the sample names in heatmap plots.

`col=c("#0080ff", "#ff00ff", "darkgreen", "#ff0000", "orange", "#00ff00", "brown")`: Character or integer vector. The color used for all line and symbol elements, unless there is a more specific control defined elsewhere. Unless groups are specified, only the first color in the vector is usually regarded.

`collapse=FALSE`: Logical scalar. Collapse overlapping ranges and aggregate the underlying data.

`degree=1`: Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See [lattice::panel.loess](#) for details.

`do.out=TRUE`: Logical scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

`evaluation=50`: Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See [lattice::panel.loess](#) for details.

`factor=0.5`: Numeric scalar. Factor to control amount of jittering in xy-type plots. See [lattice::panel.xyplot](#) for details.

`family="symmetric"`: Character scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See [lattice::panel.loess](#) for details.

`fill.confint=NULL`: Character vector. Fill colors for the confidence intervals for confint-type plots.

`fill.histogram=NULL`: Character scalar. Fill color in histogram-type plots, defaults to the setting of `fill`.

`fill.horizon=c("#B41414", "#E03231", "#F7A99C", "#9FC8DC", "#468CC8", "#0165B3")`: The fill colors for the segments in the horizon-type plot. This should be a vector of length six, where the first three entries are the colors for positive changes, and the latter three entries are the colors for negative changes. Defaults to a red-blue color scheme. See [latticeExtra::panel.horizonplot](#) for details.

`fill.mountain=c("#CCFFFF", "#FFCCFF")`: Character vector of length 2. Fill color in mountain-type and polygon-type plots.

`fontface.legend=NULL`: Integer or character scalar. The font face for the legend text.

`fontfamily.legend=NULL`: Integer or character scalar. The font family for the legend text.

`fontsize.legend=NULL`: Numeric scalar. The pixel size for the legend text.

`fontcolor.legend="#808080"`: Integer or character scalar. The font color for the legend text.

`gradient=c("#F7FBFF", "#DEEBF7", "#C6DBEF", "#9ECAE1", "#6BAED6", "#4292C6", "#2171B5", "#08519C")`: Character vector. The base colors for the gradient plotting type or the heatmap type with a single group. When plotting heatmaps with more than one group, the `col` parameter can be used to control the group color scheme, however the gradient will always be from white to 'col' and thus does not offer as much flexibility as this gradient parameter.

groups=NULL: Vector coercable to a factor. Optional sample grouping. See 'Details' section in [DataTrack](#) for further information.

horizon.origin=0: The baseline relative to which changes are indicated on the horizon-type plot. See [latticeExtra::panel.horizonplot](#) for details.

horizon.scale=NULL: The scale for each of the segments in the horizon-type plot. Defaults to 1/3 of the absolute data range. See [latticeExtra::panel.horizonplot](#) for details.

jitter.x=FALSE: Logical scalar. Toggle on jittering on the x axis in xy-type plots. See [lattice::panel.xyplot](#) for details.

jitter.y=FALSE: Logical scalar. Toggle off jittering on the y axis in xy-type plots. See [lattice::panel.xyplot](#) for details.

levels.fos=NULL: Numeric scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

legend=TRUE: Boolean triggering the addition of a legend to the track to indicate groups. This only has an effect if at least two groups are present.

lineheight.legend=NULL: Numeric scalar. The line height for the legend text.

lty.baseline=NULL: Character or numeric scalar. Line type of the optional baseline, defaults to the setting of `lty`.

lty.mountain=NULL: Character or numeric scalar. Line type in mountain-type and polygon-type plots, defaults to the setting of `lty`.

lwd.baseline=NULL: Numeric scalar. Line width of the optional baseline, defaults to the setting of `lwd`.

lwd.mountain=NULL: Numeric scalar. Line width in mountain-type and polygon-type plots, defaults to the setting of `lwd`.

min.distance=0: Numeric scalar. The minimum distance in pixel below which to collapse ranges.

na.rm=FALSE: Boolean controlling whether to discard all NA values when plotting or to keep empty spaces for NAs

ncolor=100: Integer scalar. The number of colors for the 'gradient' plotting type

notch.frac=0.5: Numeric scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

notch=FALSE: Logical scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

pch=20: Integer scalar. The type of glyph used for plotting symbols.

separator=0: Numeric scalar. Number of pixels used to separate individual samples in heatmap- and horizon-type plots.

showColorBar=TRUE: Boolean. Indicate the data range color mapping in the axis for 'heatmap' or 'gradient' types.

showSampleNames=FALSE: Boolean. Display the names of the individual samples in a heatmap or a horizon plot.

size=NULL: Numeric scalar. The relative size of the track. Can be overridden in the [plotTracks](#) function. By default the size will be set automatically based on the selected plotting type.

span=0.2: Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See [lattice::panel.loess](#) for details.

stackedBars=TRUE: Logical scalar. When there are several data groups, draw the histogram-type plots as stacked barplots or grouped side by side.

`stats=X[[i]]`: Function. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

`transformation=NULL`: Function. Applied to the data matrix prior to plotting or when calling the score method. The function should accept exactly one input argument and its return value needs to be a numeric vector which can be coerced back into a data matrix of identical dimensionality as the input data.

`type="p"`: Character vector. The plot type, one or several in `p, l, b, a, a_confint, s, g, r, S, confint, smooth, histogram, mountain, polygon, h, boxplot, gradient, heatmap, horizon`. See 'Details' section in [DataTrack](#) for more information on the individual plotting types.

`varwidth=FALSE`: Logical scalar. Parameter controlling the boxplot appearance. See [lattice::panel.bwplot](#) for details.

`window=NULL`: Numeric or character scalar. Aggregate the rows values of the data matrix to window equally sized slices on the data range using the method defined in aggregation. If negative, apply a running window of size `windowSize` using the same aggregation method. Alternatively, the special value `auto` causes the function to determine the optimal window size to avoid overplotting, and `fixed` uses fixed-size windows of size `windowSize`.

`windowSize=NULL`: Numeric scalar. The size of the running window when the value of window is negative.

`ylim=NULL`: Numeric vector of length 2. The range of the y-axis scale.

`yTicksAt=NULL`: Numeric vector. The points at which y-axis tick-marks are to be drawn. By default, when `NULL`, tickmark locations are computed.

Inherited from class GdObject:

`alpha=1`: Numeric scalar. The transparency for all track items.

`alpha.title=NULL`: Numeric scalar. The transparency for the title panel.

`background.panel="transparent"`: Integer or character scalar. The background color of the content panel.

`background.title="lightgray"`: Integer or character scalar. The background color for the title panel.

`background.legend="transparent"`: Integer or character scalar. The background color for the legend.

`cex.axis=NULL`: Numeric scalar. The expansion factor for the axis annotation. Defaults to `NULL`, in which case it is automatically determined based on the available space.

`cex.title=NULL`: Numeric scalar. The expansion factor for the title panel. This effects the fontsize of both the title and the axis, if any. Defaults to `NULL`, which means that the text size is automatically adjusted to the available space.

`col.axis="white"`: Integer or character scalar. The font and line color for the y axis, if any.

`col.border.title="white"`: Integer or character scalar. The border color for the title panels.

`col.frame="lightgray"`: Integer or character scalar. The line color used for the panel frame, if `frame==TRUE`

`col.grid="#808080"`: Integer or character scalar. Default line color for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`col.line=NULL`: Integer or character scalar. Default colors for plot lines. Usually the same as the global `col` parameter.

`col.symbol=NULL`: Integer or character scalar. Default colors for plot symbols. Usually the same as the global `col` parameter.

`col.title="white"`: Integer or character scalar. The border color for the title panels

`fill="lightgray"`: Integer or character scalar. Default fill color setting for all plotting elements, unless there is a more specific control defined elsewhere.

`fontcolor="black"`: Integer or character scalar. The font color for all text, unless a more specific definition exists.

`fontface.title=2`: Integer or character scalar. The font face for the title panels.

`fontface=1`: Integer or character scalar. The font face for all text, unless a more specific definition exists.

`fontfamily.title="sans"`: Integer or character scalar. The font family for the title panels.

`fontfamily="sans"`: Integer or character scalar. The font family for all text, unless a more specific definition exists.

`fontsize=12`: Numeric scalar. The font size for all text, unless a more specific definition exists.

`frame=FALSE`: Boolean. Draw a frame around the track when plotting.

`h=-1`: Integer scalar. Parameter controlling the number of horizontal grid lines, see [lattice::panel.grid](#) for details.

`lineheight=1`: Numeric scalar. The font line height for all text, unless a more specific definition exists.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`lty="solid"`: Numeric scalar. Default line type setting for all plotting elements, unless there is a more specific control defined elsewhere.

`lwd.border.title=1`: Integer scalar. The border width for the title panels.

`lwd.title=1`: Integer scalar. The border width for the title panels

`lwd.grid=1`: Numeric scalar. Default line width for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`lwd=1`: Numeric scalar. Default line width setting for all plotting elements, unless there is a more specific control defined elsewhere.

`min.height=3`: Numeric scalar. The minimum range height in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`min.width=1`: Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`reverseStrand=FALSE`: Logical scalar. Set up the plotting coordinates in 3' -> 5' direction if TRUE. This will effectively mirror the plot on the vertical axis.

`rotation.title=90`: The rotation angle for the text in the title panel. Even though this can be adjusted, the automatic resizing of the title panel will currently not work, so use at own risk.

`rotation=0`: The rotation angle for all text unless a more specific definition exists.

`showAxis=TRUE`: Boolean controlling whether to plot a y axis (only applies to track types where axes are implemented).

`showTitle=TRUE`: Boolean controlling whether to plot a title panel. Although this can be set individually for each track, in multi-track plots as created by [plotTracks](#) there will still be an empty placeholder in case any of the other tracks include a title. The same holds

true for axes. Note that the the title panel background color could be set to transparent in order to completely hide it.

`v=-1`: Integer scalar. Parameter controlling the number of vertical grid lines, see [lattice::panel.grid](#) for details.

IdeogramTrack: `background.title="transparent"`: Character scalar. The background color for the title panel. Defaults to omit the background.

`bevel=0.45`: Numeric scalar, between 0 and 1. The level of smoothness for the two ends of the ideogram.

`centromereShape="triangle"`: Character scalar. The shape of the centromere. Only "triangle" or "circle" is accepted. Default to "triangle"

`cex.bands=0.7`: Numeric scalar. The font expansion factor for the chromosome band identifier text.

`cex=0.8`: Numeric scalar. The overall font expansion factor for the chromosome name text.

`col="red"`: Character scalar. The border color used for the highlighting of the currently displayed genomic region.

`col.border.title="transparent"`: Integer or character scalar. The border color for the title panels.

`lwd.border.title=1`: Integer scalar. The border width for the title panels.

`fill="#FFE3E6"`: Character scalar. The fill color used for the highlighting of the currently displayed genomic region.

`fontface=1`: Character scalar. The font face for the chromosome name text.

`fontfamily="sans"`: Character scalar. The font family for the chromosome name text.

`fontcolor="#808080"`: Character scalar. The font color for the chromosome name text.

`fontsize=10`: Numeric scalar. The font size for the chromosome name text.

`outline=FALSE`: Logical scalar. Add borders to the individual chromosome staining bands.

`showBandId=FALSE`: Logical scalar. Show the identifier for the chromosome bands if there is space for it.

`lty=1`: Character or integer scalar. The line type used for the highlighting of the currently displayed genomic region.

`lwd=1`: Numeric scalar. The line width used for the highlighting of the currently displayed genomic region.

`showId=TRUE`: Logical scalar. Indicate the chromosome name next to the ideogram.

`showTitle=FALSE`: Logical scalar. Plot a title panel. Defaults to omit the title panel.

`size=NULL`: Numeric scalar. The relative size of the track. Defaults to automatic size setting.

Can also be overridden in the [plotTracks](#) function.

Inherited from class GdObject:

`alpha=1`: Numeric scalar. The transparency for all track items.

`alpha.title=NULL`: Numeric scalar. The transparency for the title panel.

`background.panel="transparent"`: Integer or character scalar. The background color of the content panel.

`background.legend="transparent"`: Integer or character scalar. The background color for the legend.

`cex.axis=NULL`: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is automatically determined based on the available space.

`cex.title=NULL`: Numeric scalar. The expansion factor for the title panel. This effects the fontsize of both the title and the axis, if any. Defaults to `NULL`, which means that the text size is automatically adjusted to the available space.

`col.axis="white"`: Integer or character scalar. The font and line color for the y axis, if any.

`col.frame="lightgray"`: Integer or character scalar. The line color used for the panel frame, if `frame==TRUE`

`col.grid="#808080"`: Integer or character scalar. Default line color for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`col.line=NULL`: Integer or character scalar. Default colors for plot lines. Usually the same as the global `col` parameter.

`col.symbol=NULL`: Integer or character scalar. Default colors for plot symbols. Usually the same as the global `col` parameter.

`col.title="white"`: Integer or character scalar. The border color for the title panels

`collapse=TRUE`: Boolean controlling whether to collapse the content of the track to accommodate the minimum current device resolution. See [collapsing](#) for details.

`fontface.title=2`: Integer or character scalar. The font face for the title panels.

`fontfamily.title="sans"`: Integer or character scalar. The font family for the title panels.

`frame=FALSE`: Boolean. Draw a frame around the track when plotting.

`grid=FALSE`: Boolean, switching on/off the plotting of a grid.

`h=-1`: Integer scalar. Parameter controlling the number of horizontal grid lines, see [lattice::panel.grid](#) for details.

`lineheight=1`: Numeric scalar. The font line height for all text, unless a more specific definition exists.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`lwd.title=1`: Integer scalar. The border width for the title panels

`lwd.grid=1`: Numeric scalar. Default line width for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`min.distance=1`: Numeric scalar. The minimum pixel distance before collapsing range items, only if `collapse==TRUE`. See [collapsing](#) for details.

`min.height=3`: Numeric scalar. The minimum range height in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`min.width=1`: Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`reverseStrand=FALSE`: Logical scalar. Set up the plotting coordinates in 3' -> 5' direction if `TRUE`. This will effectively mirror the plot on the vertical axis.

`rotation.title=90`: The rotation angle for the text in the title panel. Even though this can be adjusted, the automatic resizing of the title panel will currently not work, so use at own risk.

`rotation=0`: The rotation angle for all text unless a more specific definition exists.

`showAxis=TRUE`: Boolean controlling whether to plot a y axis (only applies to track types where axes are implemented).

`v=-1`: Integer scalar. Parameter controlling the number of vertical grid lines, see [lattice::panel.grid](#) for details.

AnnotationTrack: arrowHeadWidth=30: Numeric scalar. The width of the arrow head in pixels if shape is fixedArrow.

arrowHeadMaxWidth=40: Numeric scalar. The maximum width of the arrow head in pixels if shape is arrow.

cex.group=0.6: Numeric scalar. The font expansion factor for the group-level annotation.

cex=1: Numeric scalar. The font expansion factor for item identifiers.

col.line="darkgray": Character scalar. The color used for connecting lines between grouped items. Defaults to a light gray, but if set to NULL the same color as for the first item in the group is used.

col="transparent": Character or integer scalar. The border color for all track items.

featureAnnotation=NULL: Character scalar. Add annotation information to the individual track elements. This can be a value in id, group or feature. Defaults to id. Only works if showFeatureId is not FALSE.

fill="lightblue": Character or integer scalar. The fill color for untyped items. This is also used to connect grouped items. See [grouping](#) for details.

fontfamily.group="sans": Character scalar. The font family for the group-level annotation.

fontcolor.group="#808080": Character or integer scalar. The font color for the group-level annotation.

fontcolor.item="white": Character or integer scalar. The font color for item identifiers.

fontface.group=2: Numeric scalar. The font face for the group-level annotation.

fontsize.group=12: Numeric scalar. The font size for the group-level annotation.

groupAnnotation=NULL: Character scalar. Add annotation information as group labels. This can be a value in id, group or feature. Defaults to group. Only works if showId is not FALSE.

just.group="left": Character scalar. the justification of group labels. Either left, right, above or below.

lex=1: Numeric scalar. The line expansion factor for all track items. This is also used to connect grouped items. See [grouping](#) for details.

lineheight=1: Numeric scalar. The font line height for item identifiers.

lty="solid": Character or integer scalar. The line type for all track items. This is also used to connect grouped items. See [grouping](#) for details.

lwd=1: Integer scalar. The line width for all track items. This is also used to connect grouped items. See [grouping](#) for details.

mergeGroups=FALSE: Logical scalar. Merge fully overlapping groups if collapse==TRUE.

min.height=3: Numeric scalar. The minimum range height in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details. For feathered bars indicating the strandedness of grouped items this also controls the height of the arrow feathers.

min.width=1: Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

rotation=0: Numeric scalar. The degree of text rotation for item identifiers.

rotation.group=0: Numeric scalar. The degree of text rotation for group labels.

rotation.item=0: Numeric scalar. The degree of text rotation for item identifiers.

`shape="arrow"`: Character scalar. The shape in which to display the track items. Currently only box, arrow, fixedArrow, ellipse, and smallArrow are implemented.

`showFeatureId=FALSE`: Logical scalar. Control whether to plot the individual track item identifiers.

`showId=FALSE`: Logical scalar. Control whether to annotate individual groups.

`showOverplotting=FALSE`: Logical scalar. Use a color gradient to show the amount of overplotting for collapsed items. This implies that `collapse==TRUE`

`size=1`: Numeric scalar. The relative size of the track. Can be overridden in the [plotTracks](#) function.

Inherited from class StackedTrack:

`stackHeight=0.75`: Numeric between 0 and 1. Controls the vertical size and spacing between stacked elements. The number defines the proportion of the total available space for the stack that is used to draw the glyphs. E.g., a value of 0.5 means that half of the available vertical drawing space (for each stacking line) is used for the glyphs, and thus one quarter of the available space each is used for spacing above and below the glyph. Defaults to 0.75.

`reverseStacking=FALSE`: Logical flag. Reverse the y-ordering of stacked items. I.e., features that are plotted on the bottom-most stacks will be moved to the top-most stack and vice versa.

Inherited from class GdObject:

`alpha=1`: Numeric scalar. The transparency for all track items.

`alpha.title=NULL`: Numeric scalar. The transparency for the title panel.

`background.panel="transparent"`: Integer or character scalar. The background color of the content panel.

`background.title="lightgray"`: Integer or character scalar. The background color for the title panel.

`background.legend="transparent"`: Integer or character scalar. The background color for the legend.

`cex.axis=NULL`: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is automatically determined based on the available space.

`cex.title=NULL`: Numeric scalar. The expansion factor for the title panel. This effects the fontsize of both the title and the axis, if any. Defaults to NULL, which means that the text size is automatically adjusted to the available space.

`col.axis="white"`: Integer or character scalar. The font and line color for the y axis, if any.

`col.border.title="white"`: Integer or character scalar. The border color for the title panels.

`col.frame="lightgray"`: Integer or character scalar. The line color used for the panel frame, if `frame==TRUE`

`col.grid="#808080"`: Integer or character scalar. Default line color for grid lines, both when `type=="g"` in [DataTracks](#) and when `display` parameter `grid==TRUE`.

`col.symbol=NULL`: Integer or character scalar. Default colors for plot symbols. Usually the same as the global `col` parameter.

`col.title="white"`: Integer or character scalar. The border color for the title panels

`collapse=TRUE`: Boolean controlling whether to collapse the content of the track to accommodate the minimum current device resolution. See [collapsing](#) for details.

fontcolor="black": Integer or character scalar. The font color for all text, unless a more specific definition exists.

fontface.title=2: Integer or character scalar. The font face for the title panels.

fontface=1: Integer or character scalar. The font face for all text, unless a more specific definition exists.

fontfamily.title="sans": Integer or character scalar. The font family for the title panels.

fontfamily="sans": Integer or character scalar. The font family for all text, unless a more specific definition exists.

fontsize=12: Numeric scalar. The font size for all text, unless a more specific definition exists.

frame=FALSE: Boolean. Draw a frame around the track when plotting.

grid=FALSE: Boolean, switching on/off the plotting of a grid.

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see [lattice::panel.grid](#) for details.

lty.grid="solid": Integer or character scalar. Default line type for grid lines, both when type=="g" in [DataTracks](#) and when display parameter grid==TRUE.

lwd.border.title=1: Integer scalar. The border width for the title panels.

lwd.title=1: Integer scalar. The border width for the title panels

lwd.grid=1: Numeric scalar. Default line width for grid lines, both when type=="g" in [DataTracks](#) and when display parameter grid==TRUE.

min.distance=1: Numeric scalar. The minimum pixel distance before collapsing range items, only if collapse==TRUE. See [collapsing](#) for details.

reverseStrand=FALSE: Logical scalar. Set up the plotting coordinates in 3' -> 5' direction if TRUE. This will effectively mirror the plot on the vertical axis.

rotation.title=90: The rotation angle for the text in the title panel. Even though this can be adjusted, the automatic resizing of the title panel will currently not work, so use at own risk.

showAxis=TRUE: Boolean controlling whether to plot a y axis (only applies to track types where axes are implemented).

showTitle=TRUE: Boolean controlling whether to plot a title panel. Although this can be set individually for each track, in multi-track plots as created by [plotTracks](#) there will still be an empty placeholder in case any of the other tracks include a title. The same holds true for axes. Note that the title panel background color could be set to transparent in order to completely hide it.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see [lattice::panel.grid](#) for details.

GeneRegionTrack: arrowHeadWidth=10: Numeric scalar. The width of the arrow head in pixels if shape is fixedArrow.

arrowHeadMaxWidth=20: Numeric scalar. The maximum width of the arrow head in pixels if shape is arrow.

col=NULL: Character or integer scalar. The border color for all track items. Defaults to using the same color as in fill, also taking into account different track features.

collapseTranscripts=FALSE: Logical or character scalar. Can be one in gene, longest, shortest or meta. Merge all transcripts of the same gene into one single gene model. In the case of gene (or TRUE), this will only keep the start location of the first exon

and the end location of the last exon from all transcripts of the gene. For shortest and longest, only the longest or shortest transcript model is retained. For meta, a meta-transcript containing the union of all exons is formed (essentially identical to the operation `reduce(geneModel)`).

`exonAnnotation=NULL`: Character scalar. Add annotation information to the individual exon models. This can be a value in `symbol`, `gene`, `transcript`, `exon` or `feature`. Defaults to `exon`. Only works if `showExonId` is not `FALSE`.

`fill="orange"`: Character or integer scalar. The fill color for untyped items. This is also used to connect grouped items. See [grouping](#) for details.

`min.distance=0`: Numeric scalar. The minimum pixel distance before collapsing range items, only if `collapse==TRUE`. See [collapsing](#) for details. Note that a value larger than 0 may lead to UTR regions being merged to CDS regions, which in most cases is not particularly useful.

`shape=c("smallArrow", "box")`: Character scalar. The shape in which to display the track items. Currently only `box`, `arrow`, `ellipse`, and `smallArrow` are implemented.

`showExonId=NULL`: Logical scalar. Control whether to plot the individual exon identifiers.

`thinBoxFeature=c("utr", "ncRNA", "utr3", "utr5", "3UTR", "5UTR", "miRNA", "lincRNA", "three_prime_UTR")`: Character vector. A listing of feature types that should be drawn with thin boxes. Typically those are non-coding elements.

`transcriptAnnotation=NULL`: Character scalar. Add annotation information as transcript labels. This can be a value in `symbol`, `gene`, `transcript`, `exon` or `feature`. Defaults to `symbol`. Only works if `showId` is not `FALSE`.

Inherited from class `AnnotationTrack`:

`cex.group=0.6`: Numeric scalar. The font expansion factor for the group-level annotation.

`cex=1`: Numeric scalar. The font expansion factor for item identifiers.

`col.line="darkgray"`: Character scalar. The color used for connecting lines between grouped items. Defaults to a light gray, but if set to `NULL` the same color as for the first item in the group is used.

`featureAnnotation=NULL`: Character scalar. Add annotation information to the individual track elements. This can be a value in `id`, `group` or `feature`. Defaults to `id`. Only works if `showFeatureId` is not `FALSE`.

`fontfamily.group="sans"`: Character scalar. The font family for the group-level annotation.

`fontcolor.group="#808080"`: Character or integer scalar. The font color for the group-level annotation.

`fontcolor.item="white"`: Character or integer scalar. The font color for item identifiers.

`fontface.group=2`: Numeric scalar. The font face for the group-level annotation.

`fontsize.group=12`: Numeric scalar. The font size for the group-level annotation.

`groupAnnotation=NULL`: Character scalar. Add annotation information as group labels. This can be a value in `id`, `group` or `feature`. Defaults to `group`. Only works if `showId` is not `FALSE`.

`just.group="left"`: Character scalar. the justification of group labels. Either `left`, `right`, `above` or `below`.

`lex=1`: Numeric scalar. The line expansion factor for all track items. This is also used to connect grouped items. See [grouping](#) for details.

`lineheight=1`: Numeric scalar. The font line height for item identifiers.
`lty="solid"`: Character or integer scalar. The line type for all track items. This is also used to connect grouped items. See [grouping](#) for details.
`lwd=1`: Integer scalar. The line width for all track items. This is also used to connect grouped items. See [grouping](#) for details.
`mergeGroups=FALSE`: Logical scalar. Merge fully overlapping groups if `collapse==TRUE`.
`min.height=3`: Numeric scalar. The minimum range height in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details. For feathered bars indicating the strandedness of grouped items this also controls the height of the arrow feathers.
`min.width=1`: Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.
`rotation=0`: Numeric scalar. The degree of text rotation for item identifiers.
`rotation.group=0`: Numeric scalar. The degree of text rotation for group labels.
`rotation.item=0`: Numeric scalar. The degree of text rotation for item identifiers.
`showFeatureId=FALSE`: Logical scalar. Control whether to plot the individual track item identifiers.
`showId=FALSE`: Logical scalar. Control whether to annotate individual groups.
`showOverplotting=FALSE`: Logical scalar. Use a color gradient to show the amount of overplotting for collapsed items. This implies that `collapse==TRUE`.
`size=1`: Numeric scalar. The relative size of the track. Can be overridden in the [plotTracks](#) function.

Inherited from class `StackedTrack`:

`stackHeight=0.75`: Numeric between 0 and 1. Controls the vertical size and spacing between stacked elements. The number defines the proportion of the total available space for the stack that is used to draw the glyphs. E.g., a value of 0.5 means that half of the available vertical drawing space (for each stacking line) is used for the glyphs, and thus one quarter of the available space each is used for spacing above and below the glyph. Defaults to 0.75.
`reverseStacking=FALSE`: Logical flag. Reverse the y-ordering of stacked items. I.e., features that are plotted on the bottom-most stacks will be moved to the top-most stack and vice versa.

Inherited from class `GdObject`:

`alpha=1`: Numeric scalar. The transparency for all track items.
`alpha.title=NULL`: Numeric scalar. The transparency for the title panel.
`background.panel="transparent"`: Integer or character scalar. The background color of the content panel.
`background.title="lightgray"`: Integer or character scalar. The background color for the title panel.
`background.legend="transparent"`: Integer or character scalar. The background color for the legend.
`cex.axis=NULL`: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is automatically determined based on the available space.

`cex.title=NULL`: Numeric scalar. The expansion factor for the title panel. This effects the fontsize of both the title and the axis, if any. Defaults to `NULL`, which means that the text size is automatically adjusted to the available space.

`col.axis="white"`: Integer or character scalar. The font and line color for the y axis, if any.

`col.border.title="white"`: Integer or character scalar. The border color for the title panels.

`col.frame="lightgray"`: Integer or character scalar. The line color used for the panel frame, if `frame==TRUE`

`col.grid="#808080"`: Integer or character scalar. Default line color for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`col.symbol=NULL`: Integer or character scalar. Default colors for plot symbols. Usually the same as the global `col` parameter.

`col.title="white"`: Integer or character scalar. The border color for the title panels

`collapse=TRUE`: Boolean controlling whether to collapse the content of the track to accommodate the minimum current device resolution. See [collapsing](#) for details.

`fontcolor="black"`: Integer or character scalar. The font color for all text, unless a more specific definition exists.

`fontface.title=2`: Integer or character scalar. The font face for the title panels.

`fontface=1`: Integer or character scalar. The font face for all text, unless a more specific definition exists.

`fontfamily.title="sans"`: Integer or character scalar. The font family for the title panels.

`fontfamily="sans"`: Integer or character scalar. The font family for all text, unless a more specific definition exists.

`fontsize=12`: Numeric scalar. The font size for all text, unless a more specific definition exists.

`frame=FALSE`: Boolean. Draw a frame around the track when plotting.

`grid=FALSE`: Boolean, switching on/off the plotting of a grid.

`h=-1`: Integer scalar. Parameter controlling the number of horizontal grid lines, see [lattice::panel.grid](#) for details.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`lwd.border.title=1`: Integer scalar. The border width for the title panels.

`lwd.title=1`: Integer scalar. The border width for the title panels

`lwd.grid=1`: Numeric scalar. Default line width for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`reverseStrand=FALSE`: Logical scalar. Set up the plotting coordinates in 3' -> 5' direction if `TRUE`. This will effectively mirror the plot on the vertical axis.

`rotation.title=90`: The rotation angle for the text in the title panel. Even though this can be adjusted, the automatic resizing of the title panel will currently not work, so use at own risk.

`showAxis=TRUE`: Boolean controlling whether to plot a y axis (only applies to track types where axes are implemented).

`showTitle=TRUE`: Boolean controlling whether to plot a title panel. Although this can be set individually for each track, in multi-track plots as created by [plotTracks](#) there will still be an empty placeholder in case any of the other tracks include a title. The same holds

true for axes. Note that the the title panel background color could be set to transparent in order to completely hide it.

`v=-1`: Integer scalar. Parameter controlling the number of vertical grid lines, see [lattice::panel.grid](#) for details.

BiomartGeneRegionTrack: `C_segment="burlywood4"`: Character or integer scalar. Fill color for annotation objects of type 'C_segment'.

`D_segment="lightblue"`: Character or integer scalar. Fill color for annotation objects of type 'C_segment'.

`J_segment="dodgerblue2"`: Character or integer scalar. Fill color for annotation objects of type 'C_segment'.

`Mt_rRNA="yellow"`: Character or integer scalar. Fill color for annotation objects of type 'Mt_rRNA'.

`Mt_tRNA="darkgoldenrod"`: Character or integer scalar. Fill color for annotation objects of type 'Mt_tRNA'.

`Mt_tRNA_pseudogene="darkgoldenrod1"`: Character or integer scalar. Fill color for annotation objects of type 'Mt_tRNA_pseudogene'.

`V_segment="aquamarine"`: Character or integer scalar. Fill color for annotation objects of type 'V_segment'.

`miRNA="cornflowerblue"`: Character or integer scalar. Fill color for annotation objects of type 'L_segment'.

`miRNA_pseudogene="cornsilk"`: Character or integer scalar. Fill color for annotation objects of type 'miRNA_pseudogene'.

`misc_RNA="cornsilk3"`: Character or integer scalar. Fill color for annotation objects of type 'misc_RNA'.

`misc_RNA_pseudogene="cornsilk4"`: Character or integer scalar. Fill color for annotation objects of type 'misc_RNA_pseudogene'.

`protein_coding="#FFD58A"`: Character or integer scalar. Fill color for annotation objects of type 'protein_coding'.

`pseudogene="brown1"`: Character or integer scalar. Fill color for annotation objects of type 'pseudogene'.

`rRNA="darkolivegreen1"`: Character or integer scalar. Fill color for annotation objects of type 'rRNA'.

`rRNA_pseudogene="darkolivegreen"`: Character or integer scalar. Fill color for annotation objects of type 'rRNA_pseudogene'.

`retrotransposed="blueviolet"`: Character or integer scalar. Fill color for annotation objects of type 'retrotransposed'.

`scRNA="gold4"`: Character or integer scalar. Fill color for annotation objects of type 'scRNA'.

`scRNA_pseudogene="darkorange2"`: Character or integer scalar. Fill color for annotation objects of type 'scRNA_pseudogene'.

`snRNA="coral"`: Character or integer scalar. Fill color for annotation objects of type 'snRNA'.

`snRNA_pseudogene="coral3"`: Character or integer scalar. Fill color for annotation objects of type 'snRNA_pseudogene'.

`snoRNA="cyan"`: Character or integer scalar. Fill color for annotation objects of type 'snoRNA'.

`snoRNA_pseudogene="cyan2"`: Character or integer scalar. Fill color for annotation objects of type 'snoRNA_pseudogene'.

tRNA_pseudogene="antiquewhite3": Character or integer scalar. Fill color for annotation objects of type 'tRNA_pseudogene'.

utr3="#FFD58A": Character or integer scalar. Fill color for annotation objects of type 'utr3'.

utr5="#FFD58A": Character or integer scalar. Fill color for annotation objects of type 'utr5'.

verbose=FALSE: Logical scalar. Report data loading events from Bioamart or retrieval from cache.

Inherited from class GeneRegionTrack:

arrowHeadWidth=10: Numeric scalar. The width of the arrow head in pixels if shape is fixedArrow.

arrowHeadMaxWidth=20: Numeric scalar. The maximum width of the arrow head in pixels if shape is arrow.

col=NULL: Character or integer scalar. The border color for all track items. Defaults to using the same color as in fill, also taking into account different track features.

collapseTranscripts=FALSE: Logical or character scalar. Can be one in gene, longest, shortest or meta. Merge all transcripts of the same gene into one single gene model. In the case of gene (or TRUE), this will only keep the start location of the first exon and the end location of the last exon from all transcripts of the gene. For shortest and longest, only the longest or shortest transcript model is retained. For meta, a meta-transcript containing the union of all exons is formed (essentially identical to the operation `reduce(geneModel)`).

exonAnnotation=NULL: Character scalar. Add annotation information to the individual exon models. This can be a value in symbol, gene, transcript, exon or feature. Defaults to exon. Only works if showExonId is not FALSE.

fill="orange": Character or integer scalar. The fill color for untyped items. This is also used to connect grouped items. See [grouping](#) for details.

min.distance=0: Numeric scalar. The minimum pixel distance before collapsing range items, only if collapse==TRUE. See [collapsing](#) for details. Note that a value larger than 0 may lead to UTR regions being merged to CDS regions, which in most cases is not particularly useful.

shape=c("smallArrow", "box"): Character scalar. The shape in which to display the track items. Currently only box, arrow, ellipse, and smallArrow are implemented.

showExonId=NULL: Logical scalar. Control whether to plot the individual exon identifiers.

thinBoxFeature=c("utr", "ncRNA", "utr3", "utr5", "3UTR", "5UTR", "miRNA", "lincRNA", "three_prime_UTR"): Character vector. A listing of feature types that should be drawn with thin boxes. Typically those are non-coding elements.

transcriptAnnotation=NULL: Character scalar. Add annotation information as transcript labels. This can be a value in symbol, gene, transcript, exon or feature. Defaults to symbol. Only works if showId is not FALSE.

Inherited from class AnnotationTrack:

cex.group=0.6: Numeric scalar. The font expansion factor for the group-level annotation.

cex=1: Numeric scalar. The font expansion factor for item identifiers.

col.line="darkgray": Character scalar. The color used for connecting lines between grouped items. Defaults to a light gray, but if set to NULL the same color as for the first item in the group is used.

`featureAnnotation=NULL`: Character scalar. Add annotation information to the individual track elements. This can be a value in `id`, `group` or `feature`. Defaults to `id`. Only works if `showFeatureId` is not `FALSE`.

`fontfamily.group="sans"`: Character scalar. The font family for the group-level annotation.

`fontcolor.group="#808080"`: Character or integer scalar. The font color for the group-level annotation.

`fontcolor.item="white"`: Character or integer scalar. The font color for item identifiers.

`fontface.group=2`: Numeric scalar. The font face for the group-level annotation.

`fontsize.group=12`: Numeric scalar. The font size for the group-level annotation.

`groupAnnotation=NULL`: Character scalar. Add annotation information as group labels. This can be a value in `id`, `group` or `feature`. Defaults to `group`. Only works if `showId` is not `FALSE`.

`just.group="left"`: Character scalar. the justification of group labels. Either `left`, `right`, `above` or `below`.

`lex=1`: Numeric scalar. The line expansion factor for all track items. This is also used to connect grouped items. See [grouping](#) for details.

`lineheight=1`: Numeric scalar. The font line height for item identifiers.

`lty="solid"`: Character or integer scalar. The line type for all track items. This is also used to connect grouped items. See [grouping](#) for details.

`lwd=1`: Integer scalar. The line width for all track items. This is also used to connect grouped items. See [grouping](#) for details.

`mergeGroups=FALSE`: Logical scalar. Merge fully overlapping groups if `collapse==TRUE`.

`min.height=3`: Numeric scalar. The minimum range height in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details. For feathered bars indicating the strandedness of grouped items this also controls the height of the arrow feathers.

`min.width=1`: Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`rotation=0`: Numeric scalar. The degree of text rotation for item identifiers.

`rotation.group=0`: Numeric scalar. The degree of text rotation for group labels.

`rotation.item=0`: Numeric scalar. The degree of text rotation for item identifiers.

`showFeatureId=FALSE`: Logical scalar. Control whether to plot the individual track item identifiers.

`showId=FALSE`: Logical scalar. Control whether to annotate individual groups.

`showOverplotting=FALSE`: Logical scalar. Use a color gradient to show the amount of overplotting for collapsed items. This implies that `collapse==TRUE`

`size=1`: Numeric scalar. The relative size of the track. Can be overridden in the [plotTracks](#) function.

Inherited from class `StackedTrack`:

`stackHeight=0.75`: Numeric between 0 and 1. Controls the vertical size and spacing between stacked elements. The number defines the proportion of the total available space for the stack that is used to draw the glyphs. E.g., a value of 0.5 means that half of the available vertical drawing space (for each stacking line) is used for the glyphs, and thus one quarter of the available space each is used for spacing above and below the glyph. Defaults to 0.75.

`reverseStacking=FALSE`: Logical flag. Reverse the y-ordering of stacked items. I.e., features that are plotted on the bottom-most stacks will be moved to the top-most stack and vice versa.

Inherited from class `GdObject`:

`alpha=1`: Numeric scalar. The transparency for all track items.

`alpha.title=NULL`: Numeric scalar. The transparency for the title panel.

`background.panel="transparent"`: Integer or character scalar. The background color of the content panel.

`background.title="lightgray"`: Integer or character scalar. The background color for the title panel.

`background.legend="transparent"`: Integer or character scalar. The background color for the legend.

`cex.axis=NULL`: Numeric scalar. The expansion factor for the axis annotation. Defaults to `NULL`, in which case it is automatically determined based on the available space.

`cex.title=NULL`: Numeric scalar. The expansion factor for the title panel. This effects the fontsize of both the title and the axis, if any. Defaults to `NULL`, which means that the text size is automatically adjusted to the available space.

`col.axis="white"`: Integer or character scalar. The font and line color for the y axis, if any.

`col.border.title="white"`: Integer or character scalar. The border color for the title panels.

`col.frame="lightgray"`: Integer or character scalar. The line color used for the panel frame, if `frame==TRUE`

`col.grid="#808080"`: Integer or character scalar. Default line color for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`col.symbol=NULL`: Integer or character scalar. Default colors for plot symbols. Usually the same as the global `col` parameter.

`col.title="white"`: Integer or character scalar. The border color for the title panels

`collapse=TRUE`: Boolean controlling whether to collapse the content of the track to accommodate the minimum current device resolution. See [collapsing](#) for details.

`fontcolor="black"`: Integer or character scalar. The font color for all text, unless a more specific definition exists.

`fontface.title=2`: Integer or character scalar. The font face for the title panels.

`fontface=1`: Integer or character scalar. The font face for all text, unless a more specific definition exists.

`fontfamily.title="sans"`: Integer or character scalar. The font family for the title panels.

`fontfamily="sans"`: Integer or character scalar. The font family for all text, unless a more specific definition exists.

`fontsize=12`: Numeric scalar. The font size for all text, unless a more specific definition exists.

`frame=FALSE`: Boolean. Draw a frame around the track when plotting.

`grid=FALSE`: Boolean, switching on/off the plotting of a grid.

`h=-1`: Integer scalar. Parameter controlling the number of horizontal grid lines, see [lattice::panel.grid](#) for details.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`lwd.border.title=1`: Integer scalar. The border width for the title panels.
`lwd.title=1`: Integer scalar. The border width for the title panels
`lwd.grid=1`: Numeric scalar. Default line width for grid lines, both when `type=="g"` in [DataTracks](#) and when `display` parameter `grid==TRUE`.
`reverseStrand=FALSE`: Logical scalar. Set up the plotting coordinates in 3' -> 5' direction if `TRUE`. This will effectively mirror the plot on the vertical axis.
`rotation.title=90`: The rotation angle for the text in the title panel. Even though this can be adjusted, the automatic resizing of the title panel will currently not work, so use at own risk.
`showAxis=TRUE`: Boolean controlling whether to plot a y axis (only applies to track types where axes are implemented).
`showTitle=TRUE`: Boolean controlling whether to plot a title panel. Although this can be set individually for each track, in multi-track plots as created by [plotTracks](#) there will still be an empty placeholder in case any of the other tracks include a title. The same holds true for axes. Note that the title panel background color could be set to transparent in order to completely hide it.
`v=-1`: Integer scalar. Parameter controlling the number of vertical grid lines, see [lattice::panel.grid](#) for details.

AlignmentsTrack: `alpha.reads=0.5`: Numeric scalar between 0 and 1. The transparency of the individual read icons. Can be used to indicate overlapping regions in read pairs. Only on supported devices.

`alpha.mismatch=1`: Numeric scalar between 0 and 1. The transparency of the mismatch base information.
`cex=0.7`: Numeric Scalar. The global character expansion factor.
`cex.mismatch=NULL`: Numeric Scalar. The character expansion factor for the mismatch base letters.
`col.coverage=NULL`: Integer or character scalar. The line color for the coverage profile.
`col.gap="#808080"`: Integer or character scalar. The color of the line that is bridging the gap regions in gapped alignments.
`col.mates="#E0E0E0"`: Integer or character scalar. The color of the line that is connecting two paired reads.
`col.deletion="#000000"`: Integer or character scalar. The color of the line that is bridging the deleted regions in alignments.
`col.insertion="#984EA3"`: Integer or character scalar. The color of the line that highlighting insertions in alignments.
`col.mismatch="#808080"`: Integer or character scalar. The box color around mismatch bases.
`col.reads=NULL`: Integer or character scalar. The box color around reads.
`col.sashimi=NULL`: Integer or character scalar. The line color for sashimi plots.
`col="#808080"`: Integer or character scalar. The default color of all line elements.
`collapse=FALSE`: Logical scalar. Do not perform any collapsing of overlapping elements. Currently not supported.
`coverageHeight=0.1`: Numeric scalar. The height of the coverage region of the track. Can either be a value between 0 and 1 in which case it is taken as a relative height, or a positive value greater 1 in which case it is interpreted as pixels.

`fill.coverage=NULL`: Integer or character scalar. The fill color for the coverage profile.
`fill.reads=NULL`: Integer or character scalar. The fill color for the read icons.
`fill="#BABABA"`: Integer or character scalar. The default fill color of all plot elements.
`fontface.mismatch=2`: Integer scalar. The font face for mismatch bases.
`lty.coverage=NULL`: Integer or character scalar. The line type of the coverage profile.
`lty.gap=NULL`: Integer or character scalar. The type of the line that is bridging the gap regions in gapped alignments.
`lty.mates=NULL`: Integer or character scalar. The type of the line that is connecting two paired reads.
`lty.deletion=NULL`: Integer or character scalar. The type of the line that is bridging the deleted regions in alignments.
`lty.insertion=NULL`: Integer or character scalar. The type of the line that highlighting insertions in alignments.
`lty.mismatch=NULL`: Integer or character scalar. The box line type around mismatch bases.
`lty.reads=NULL`: Integer or character scalar. The box line type around mismatch reads.
`lty=1`: Integer or character scalar. The default type of all line elements.
`lwd.coverage=NULL`: Integer or character scalar. The line width of the coverage profile.
`lwd.gap=NULL`: Integer scalar. The width of the line that is bridging the gap regions in gapped alignments.
`lwd.mates=NULL`: Integer scalar. The width of the line that is connecting two paired reads.
`lwd.deletion=NULL`: Integer scalar. The width of the line that is bridging the deleted regions in alignments.
`lwd.insertion=NULL`: Integer scalar. The width of the line that highlighting insertions in alignments.
`lwd.mismatch=NULL`: Integer scalar. The box line width around mismatch bases.
`lwd.reads=NULL`: Integer scalar. The box line width around reads.
`lwd.sashimiMax=10`: Integer scalar. The maximal width of the line in sashimi plots.
`lwd=1`: Integer scalar. The default width of all line elements.
`max.height=10`: Integer scalar. The maximum height of an individual read in pixels. Can be used in combination with `min.height` to control the read and stacking appearance.
`min.height=5`: Integer scalar. The minimum height of an individual read in pixels. Can be used in combination with `max.height` to control the read and stacking appearance.
`minCoverageHeight=50`: Integer scalar. The minimum height of the coverage section. Useful in combination with a relative setting of `coverageHeight`.
`minSashimiHeight=50`: Integer scalar. The minimum height of the sashimi section. Useful in combination with a relative setting of `sashimiHeight`.
`noLetters=FALSE`: Logical scalar. Always plot colored boxes for mismatch bases regardless of the available space.
`sashimiFilter=NULL`: GRanges object. Only junctions which overlap equally with `sashimiFilter` GRanges are shown. Default NULL, no filtering.
`sashimiFilterTolerance=0`: Integer scalar. Only used in combination with `sashimiFilter`. It allows to include junctions whose starts/ends are within specified distance from `sashimiFilter` GRanges. This is useful for cases where the aligner did not place the junction reads precisely. Default 0L, no tolerance.

`sashimiHeight=0.1`: Integer scalar. The height of the sashimi part of the track. Can either be a value between 0 and 1 in which case it is taken as a relative height, or a positive value greater 1 in which case it is interpreted as pixels.

`sashimiScore=1`: Integer scalar. The minimum number of reads supporting the junction.

`sashimiStrand="*"`: Integer scalar. Only reads which have the specified strand are considered to count the junctions.

`sashimiTransformation=NULL`: Function. Applied to the junction score vector prior to plotting. The function should accept exactly one input argument and its return value needs to be a numeric vector of identical length as the input data.

`showIndels=FALSE`: Logical scalar. Consider insertions and deletions in coverage and pile-up. Default is FALSE. If set to TRUE the deletions defined in CIGAR string are not considered in coverage plot. The deletions are displayed as bridging lines in pile-up track. Insertions are shown as vertical bars.

`showMismatches=TRUE`: Logical scalar. Add mismatch information, either as individual base letters or using color coded bars. This implies that the reference sequence has been provided, either to the class constructor or as part of the track list.

`size=NULL`: Numeric scalar. The size of the track. Defaults to automatic sizing.

`transformation=NULL`: Function. Applied to the coverage vector prior to plotting. The function should accept exactly one input argument and its return value needs to be a numeric Rle of identical length as the input data.

`type=c("coverage", "pileup")`: Character vector. The type of information to plot. For coverage a coverage plot, potentially augmented by base mismatch information, for sashimi a sashimi plot, showing the junctions, and for pileup the pileups of the individual reads. These three can be combined.

Inherited from class `StackedTrack`:

`stackHeight=0.75`: Numeric between 0 and 1. Controls the vertical size and spacing between stacked elements. The number defines the proportion of the total available space for the stack that is used to draw the glyphs. E.g., a value of 0.5 means that half of the available vertical drawing space (for each stacking line) is used for the glyphs, and thus one quarter of the available space each is used for spacing above and below the glyph. Defaults to 0.75.

`reverseStacking=FALSE`: Logical flag. Reverse the y-ordering of stacked items. I.e., features that are plotted on the bottom-most stacks will be moved to the top-most stack and vice versa.

Inherited from class `GdObject`:

`alpha=1`: Numeric scalar. The transparency for all track items.

`alpha.title=NULL`: Numeric scalar. The transparency for the title panel.

`background.panel="transparent"`: Integer or character scalar. The background color of the content panel.

`background.title="lightgray"`: Integer or character scalar. The background color for the title panel.

`background.legend="transparent"`: Integer or character scalar. The background color for the legend.

`cex.axis=NULL`: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is automatically determined based on the available space.

`cex.title=NULL`: Numeric scalar. The expansion factor for the title panel. This effects the fontsize of both the title and the axis, if any. Defaults to `NULL`, which means that the text size is automatically adjusted to the available space.

`col.axis="white"`: Integer or character scalar. The font and line color for the y axis, if any.

`col.border.title="white"`: Integer or character scalar. The border color for the title panels.

`col.frame="lightgray"`: Integer or character scalar. The line color used for the panel frame, if `frame==TRUE`

`col.grid="#808080"`: Integer or character scalar. Default line color for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`col.line=NULL`: Integer or character scalar. Default colors for plot lines. Usually the same as the global `col` parameter.

`col.symbol=NULL`: Integer or character scalar. Default colors for plot symbols. Usually the same as the global `col` parameter.

`col.title="white"`: Integer or character scalar. The border color for the title panels

`fontcolor="black"`: Integer or character scalar. The font color for all text, unless a more specific definition exists.

`fontface.title=2`: Integer or character scalar. The font face for the title panels.

`fontface=1`: Integer or character scalar. The font face for all text, unless a more specific definition exists.

`fontfamily.title="sans"`: Integer or character scalar. The font family for the title panels.

`fontfamily="sans"`: Integer or character scalar. The font family for all text, unless a more specific definition exists.

`fontsize=12`: Numeric scalar. The font size for all text, unless a more specific definition exists.

`frame=FALSE`: Boolean. Draw a frame around the track when plotting.

`grid=FALSE`: Boolean, switching on/off the plotting of a grid.

`h=-1`: Integer scalar. Parameter controlling the number of horizontal grid lines, see [latitude::panel.grid](#) for details.

`lineheight=1`: Numeric scalar. The font line height for all text, unless a more specific definition exists.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`lwd.border.title=1`: Integer scalar. The border width for the title panels.

`lwd.title=1`: Integer scalar. The border width for the title panels

`lwd.grid=1`: Numeric scalar. Default line width for grid lines, both when `type=="g"` in [DataTracks](#) and when display parameter `grid==TRUE`.

`min.distance=1`: Numeric scalar. The minimum pixel distance before collapsing range items, only if `collapse==TRUE`. See [collapsing](#) for details.

`min.width=1`: Numeric scalar. The minimum range width in pixels to display. All ranges are expanded to this size in order to avoid rendering issues. See [collapsing](#) for details.

`reverseStrand=FALSE`: Logical scalar. Set up the plotting coordinates in 3' -> 5' direction if `TRUE`. This will effectively mirror the plot on the vertical axis.

`rotation.title=90`: The rotation angle for the text in the title panel. Even though this can be adjusted, the automatic resizing of the title panel will currently not work, so use at own risk.

`rotation=0`: The rotation angle for all text unless a more specific definition exists.

`showAxis=TRUE`: Boolean controlling whether to plot a y axis (only applies to track types where axes are implemented).

`showTitle=TRUE`: Boolean controlling whether to plot a title panel. Although this can be set individually for each track, in multi-track plots as created by `plotTracks` there will still be an empty placeholder in case any of the other tracks include a title. The same holds true for axes. Note that the title panel background color could be set to transparent in order to completely hide it.

`v=-1`: Integer scalar. Parameter controlling the number of vertical grid lines, see `lattice::panel.grid` for details.

Author(s)

Florian Hahne

See Also

[AnnotationTrack](#)
[DataTrack](#)
[DisplayPars](#)
[GdObject](#)
[availableDisplayPars](#)
[collapsing](#)
[grouping](#)
[latticeExtra::panel.horizonplot](#)
[lattice::panel.bwplot](#)
[lattice::panel.grid](#)
[lattice::panel.loess](#)
[lattice::panel.xyplot](#)
[plotTracks](#)

Examples

```
## Which scheme is used?
getOption("Gviz.scheme")

## Change default settings for GeneRegionTrack
scheme <- getScheme()
scheme$GeneRegionTrack$fill <- "salmon"
scheme$GeneRegionTrack$col <- NULL
scheme$GeneRegionTrack$transcriptAnnotation <- "transcript"

## replace default scheme with myScheme
addScheme(scheme, "myScheme")
options(Gviz.scheme = "myScheme")
getOption("Gviz.scheme")
```

```
data(geneModels)
grtrack <- GeneRegionTrack(geneModels, genome = "hg19", chromosome = "chr7", name = "Gene Model")
plotTracks(grtrack)
```

StackedTrack-class	<i>StackedTrack class and methods</i>
--------------------	---------------------------------------

Description

The virtual parent class for all track types in the Gviz package which contain potentially overlapping annotation items that have to be stacked when plotted.

Usage

```
## S4 method for signature 'StackedTrack'
initialize(.Object, stacking, ...)

## S4 method for signature 'StackedTrack'
stacking(GdObject)

## S4 replacement method for signature 'StackedTrack,character'
stacking(GdObject) <- value

## S4 method for signature 'StackedTrack'
stacks(GdObject)

## S4 method for signature 'StackedTrack'
setStacks(GdObject, ...)

## S4 method for signature 'StackedTrack'
consolidateTrack(GdObject, ...)

## S4 method for signature 'StackedTrack,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'StackedTrack'
subset(x, from = NULL, to = NULL, sort = FALSE, stacks = FALSE, ...)

## S4 method for signature 'StackedTrack'
drawGD(GdObject, ...)
```

Arguments

.Object	.Object
stacking	stacking
...	Additional arguments.

GdObject	Object of GdObject-class.
value	Value to be set.
x	A valid track object class name, or the object itself, in which case the class is derived directly from it.
i	Numeric scalar, index to subset.
j	Numeric scalar, index to subset. Ignored.
drop	logical, indicating if levels that do not occur should be dropped (if f is a factor).
from, to	Numeric scalar, giving the range of genomic coordinates to limit the tracks in. Note that to cannot be larger than from.
sort	logical.
stacks	logical. Set if stacking should be preserved.

Value

A virtual Class: No objects may be created from it.

Functions

- `initialize(StackedTrack)`: Initialize the stacking slot (validating it against the allowed stackingValues) and reset the stacks slot, before deferring to the RangeTrack initializer for the remaining slots.
- `stacking(StackedTrack)`: return the current stacking type.
- `stacking(GdObject = StackedTrack) <- value`: set the object's stacking type to one in `c(hide, dense, squish, pack, full)`.
- `stacks(StackedTrack)`: return the stack indices for each track item.
- `setStacks(StackedTrack)`: recompute the stacks based on the available space and on the object's track items and stacking settings.
- `consolidateTrack(StackedTrack)`: For StackedTrack objects, set the stacking (which could have been passed in as a display parameter) before deferring to the RangeTrack method for the remaining consolidation steps.
- `x = StackedTrack[i = ANY, j = ANY, drop = ANY]`: subset the items in the StackedTrack object. This is essentially similar to subsetting of the GRanges object in the range slot. For most applications, the subset method may be more appropriate.
- `subset(StackedTrack)`: subset a StackedTrack by coordinates and sort if necessary.
- `drawGD(StackedTrack)`: plot the object to a graphics device. The return value of this method is the input object, potentially updated during the plotting operation. Internally, there are two modes in which the method can be called. Either in 'prepare' mode, in which case no plotting is done but the stacking information is updated based on the available space, or in 'plotting' mode, in which case the actual graphical output is created. Note that the method for this particular subclass is usually called through inheritance and not particularly useful on its own.

Slots

- dp Object of DisplayPars-class, the display settings controlling the look and feel of a track. See settings for details on setting graphical parameters for tracks.
- name Object of class character, a human-readable name for the track that will be used in the track's annotation panel if necessary.
- imageMap Object of ImageMap-class, containing optional information for an HTML image map. This will be created by the drawGD methods when the track is plotted to a device and is usually not set by the user.
- range Object of class GRanges, the genomic ranges of the track items as well as additional annotation information in its elementMetadata slot. Please note that the slot is actually implemented as a class union between GRanges and IRanges to increase efficiency, for instance for DataTrack objects. This usually does not concern the user.
- chromosome Object of class character, the chromosome on which the track is defined. There can only be a single chromosome for one track. For certain subclasses, the space of allowed chromosome names is limited (e.g., only those chromosomes that exist for a particular genome). Throughout the package, chromosome names have to be entered either as a single integer scalar or as a character scalar of the form chrXYZ, where XYZ may be an arbitrary character string.
- genome Object of class character, the genome for which the track is defined. For most sub-classes this has to be a valid UCSC genome identifier, however this may not always be formally checked upon object instantiation.
- stacking Object of class character, the stacking type of overlapping items on the final plot. One in c(hide, dense, squish, pack, full). Currently, only hide (do not show the track items at all), squish (make best use of the available space) and dense (no stacking at all) are implemented.
- stacks Object of class numeric, holding the stack indices for each track item. This slot is usually populated by calling the setStacks method upon plotting, since the correct stacking is a function of the available plotting space.

Author(s)

Florian Hahne

See Also

[DisplayPars](#)
[GdObject](#)
[GRanges](#)
[HighlightTrack](#)
[ImageMap](#)
[IRanges](#)
[RangeTrack](#)
[DataTrack](#)
[collapsing](#)

[grouping](#)
[panel.grid](#)
[plotTracks](#)
[settings](#)

Examples

```
## This is a reference class therefore we show below
## an example from AnnotationTrack

## An empty object
AnnotationTrack()

## Construct from individual arguments
st <- c(2000000, 2070000, 2100000, 2160000)
ed <- c(2050000, 2130000, 2150000, 2170000)
str <- c("-", "+", "-", "-")
gr <- c("Group1", "Group2", "Group1", "Group3")

annTrack <- AnnotationTrack(
  start = st, end = ed, strand = str, chromosome = 7,
  genome = "hg19", feature = "test", group = gr,
  id = paste("annTrack item", 1:4),
  name = "generic annotation", stacking = "squish"
)

## Plotting
plotTracks(annTrack)

## Stacking
stacking(annTrack)
stacking(annTrack) <- "dense"
plotTracks(annTrack)
```

UcscTrack

Meta-constructor for Gviz tracks fetched directly from the various UCSC data sources.

Description

The UCSC data base provides a wealth of annotation information. This function can be used to access UCSC, to retrieve the data available there and to return it as an annotation track object amenable to plotting with [plotTracks](#).

Usage

```

UcscTrack(
  track,
  table = NULL,
  trackType = c("AnnotationTrack", "GeneRegionTrack", "DataTrack", "GenomeAxisTrack"),
  genome,
  chromosome,
  name = NULL,
  from,
  to,
  ...
)

```

Arguments

track	Character, the name of the track to fetch from UCSC. To find out about available tracks please consult the online table browser at http://genome.ucsc.edu/cgi-bin/hgTables?command=start .
table	Character, the name of the table to fetch from UCSC, or NULL, in which case the default selection of tables is used. To find out about available tables for a given track please consult the online table browser at http://genome.ucsc.edu/cgi-bin/hgTables?command=start .
trackType	Character, one in c("AnnotationTrack", "GeneRegionTrack", "DataTrack", "GenomeAxisTrack"). The function will try to coerce the downloaded data in an object of this class. See below for details.
genome	Character, a valid USCS genome identifier for which to fetch the data.
chromosome	Character, a valid USCS character identifier for which to fetch the data.
name	Character, the name to use for the resulting track object.
from, to	A range of genomic locations for which to fetch data.
...	All additional named arguments are expected to be either display parameters for the resulting objects, or character scalars of column names in the downloaded UCSC data tables that are matched by name to available arguments in the respective constructor functions as defined by the trackType argument. See Details section for more information.

Details

`clearSessionCache` can be called to remove all cached items from the session which are generated when connecting with the UCSC data base.

The data stored at the UCSC data bases can be of different formats: gene or transcript model data, simple annotation features like CpG Island locations or SNPs, or numeric data like conservation or mapability. This function presents a unified API to download all kinds of data and to map them back to one of the annotation track objects defined in this package. The type of object to hold the data has to be given in the `trackType` argument, and subsequently the function passes all data on to the respective object constructor. All additional named arguments are considered to be relevant for the constructor of choice, and single character scalars are replaced by the respective

data columns in the downloaded UCSC tables if available. For instance, assuming the table for track 'foo' contains the columns 'id', 'type', 'fromLoc' and 'toLoc', giving the feature identifier, type, start end end location. In order to create an [AnnotationTrack](#) object from that data, we have to pass the additional named arguments `id="id"`, `feature="type"`, `start="fromLoc"` and `end="toLoc"` to the `UcscTrack` function. The complete function call could look like this:

```
UcscTrack(track="foo", genome="mm39", chromosome=3, from=1000, to=10000, trackType="AnnotationTrack",
id="id", feature="type", start="from", end="to")
```

To reduce the bandwidth, some caching of the UCSC connection takes place. In order to remove these cached session items, call `clearSessionCache`.

The `Gviz.ucscUrl` option controls which URL is being used to connect to UCSC. For instance, one could switch to the European UCSC mirror by calling `options(Gviz.ucscUrl = "http://genome-euro.ucsc.edu/cgi-bin/ucsc/trackTableByName?tbl=knownGene&org=mm39")`.

Value

An annotation track object as determined by `trackType`.

Author(s)

Florian Hahne

See Also

[AnnotationTrack](#)

[DataTrack](#)

[GeneRegionTrack](#)

[GenomeAxisTrack](#)

[plotTracks](#)

Examples

```
## Not run:

## Create UcscTrack for Known Genes from mm39 genome
from <- 65921878
to <- 65980988
knownGenes <- UcscTrack(
  genome = "mm39", chromosome = "chrX", track = "knownGene",
  from = from, to = to, trackType = "GeneRegionTrack",
  rstarts = "exonStarts", rends = "exonEnds", gene = "name",
  symbol = "name", transcript = "name", strand = "strand",
  fill = "#8282d2", name = "UCSC Genes"
)

## End(Not run)

## if the UCSC is not accessible load prepared object
data(ucscItems)

## knownGenes is essentially GeneRegionTrack
```

```
knownGenes

## plotting
plotTracks(knownGenes, chromosome = "chrX", from = 65920688, to = 65960068)
```

Index

- * **datasets**
 - datasets, [20](#)
 - .DisplayPars (DisplayPars-class), [22](#)
 - [, IdeogramTrack, ANY, ANY, ANY-method (IdeogramTrack-class), [41](#)
 - [, RangeTrack, ANY, ANY, ANY-method (RangeTrack-class), [50](#)
 - [, StackedTrack, ANY, ANY, ANY-method (StackedTrack-class), [90](#)
- addScheme (settings), [64](#)
- AlignmentsTrack
 - (AlignmentsTrack-class), [3](#)
- AlignmentsTrack-class, [3](#)
- AnnotationTrack, [17](#), [36](#), [65](#), [89](#), [95](#)
- as.list, DisplayPars-method (DisplayPars-class), [22](#)
- as.list, InferredDisplayPars-method (DisplayPars-class), [22](#)
- availableDefaultMapping, [10](#)
- availableDisplayPars, [65](#), [89](#)
- availableDisplayPars (DisplayPars-class), [22](#)
- axTrack (datasets), [20](#)
- biomaRt::getBM, [14](#)
- BiomartGeneRegionTrack, [36](#)
- BiomartGeneRegionTrack
 - (BiomartGeneRegionTrack-class), [12](#)
- BiomartGeneRegionTrack-class, [12](#)
- biomTrack (datasets), [20](#)
- biomTrack2 (datasets), [20](#)
- bmt (datasets), [20](#)
- bmTrack (datasets), [20](#)
- BSgenome, [60](#)
- chromosome, RangeTrack-method (RangeTrack-class), [50](#)
- chromosome, SequenceTrack-method (SequenceTrack-class), [58](#)
- chromosome<-, AlignmentsTrack-method (AlignmentsTrack-class), [3](#)
- chromosome<-, HighlightTrack-method (HighlightTrack-class), [37](#)
- chromosome<-, IdeogramTrack-method (IdeogramTrack-class), [41](#)
- chromosome<-, RangeTrack-method (RangeTrack-class), [50](#)
- chromosome<-, SequenceTrack-method (SequenceTrack-class), [58](#)
- clearSessionCache, [44](#)
- clearSessionCache (UcscTrack), [93](#)
- collapsing, [9](#), [11](#), [15](#), [17](#), [19](#), [33](#), [40](#), [44](#), [46](#), [55](#), [63](#), [67](#), [68](#), [72](#), [74–80](#), [82–84](#), [88](#), [89](#), [92](#)
- conservation (datasets), [20](#)
- consolidateTrack, HighlightTrack-method (HighlightTrack-class), [37](#)
- consolidateTrack, RangeTrack-method (RangeTrack-class), [50](#)
- consolidateTrack, SequenceTrack-method (SequenceTrack-class), [58](#)
- consolidateTrack, StackedTrack-method (StackedTrack-class), [90](#)
- cpgIslands (datasets), [20](#)
- ctrack (datasets), [20](#)
- CustomTrack (CustomTrack-class), [18](#)
- CustomTrack-class, [18](#)
- cyp2b10 (datasets), [20](#)
- datasets, [20](#)
- DataTrack, [9](#), [11](#), [15](#), [17](#), [19](#), [33](#), [40](#), [44](#), [46](#), [55](#), [63](#), [67](#), [68](#), [70–72](#), [74](#), [76](#), [77](#), [80](#), [84](#), [85](#), [88](#), [89](#), [92](#), [95](#)
- denseAnnTrack (datasets), [20](#)
- DetailsAnnotationTrack
 - (DetailsAnnotationTrack-class), [20](#)

- DetailsAnnotationTrack-class, 20
- DisplayPars, 8, 11, 15, 19, 33, 40, 44, 46, 55, 63, 65, 89, 92
- DisplayPars (DisplayPars-class), 22
- displayPars (DisplayPars-class), 22
- displayPars, DisplayPars, character-method (DisplayPars-class), 22
- displayPars, DisplayPars, missing-method (DisplayPars-class), 22
- DisplayPars-class, 22
- displayPars<- (DisplayPars-class), 22
- displayPars<-, DisplayPars, list-method (DisplayPars-class), 22
- displayPars<-, HighlightTrack, list-method (HighlightTrack-class), 37
- DNAStringSet, 60
- drawAxis, AlignmentsTrack-method (AlignmentsTrack-class), 3
- drawAxis, NumericTrack-method (NumericTrack-class), 45
- drawGD, AlignmentsTrack-method (AlignmentsTrack-class), 3
- drawGD, CustomTrack-method (CustomTrack-class), 18
- drawGD, DetailsAnnotationTrack-method (DetailsAnnotationTrack-class), 20
- drawGD, GeneRegionTrack-method (GeneRegionTrack-class), 26
- drawGD, IdeogramTrack-method (IdeogramTrack-class), 41
- drawGD, SequenceTrack-method (SequenceTrack-class), 58
- drawGD, StackedTrack-method (StackedTrack-class), 90
- drawGrid, AlignmentsTrack-method (AlignmentsTrack-class), 3
- drawGrid, NumericTrack-method (NumericTrack-class), 45
- dtHoriz (datasets), 20
- end, IdeogramTrack-method (IdeogramTrack-class), 41
- end, RangeTrack-method (RangeTrack-class), 50
- end, SequenceTrack-method (SequenceTrack-class), 58
- end<-, IdeogramTrack-method (IdeogramTrack-class), 41
- end<-, RangeTrack-method (RangeTrack-class), 50
- ensGenes (datasets), 20
- exon, GeneRegionTrack-method (GeneRegionTrack-class), 26
- exon<-, GeneRegionTrack, character-method (GeneRegionTrack-class), 26
- exportTracks, 25
- feature, RangeTrack-method (RangeTrack-class), 50
- feature<-, RangeTrack, character-method (RangeTrack-class), 50
- from (datasets), 20
- gcContent (datasets), 20
- GdObject, 8, 11, 15, 19, 33, 40, 44, 46, 47, 49, 55, 63, 64, 89, 92
- gene, GeneRegionTrack-method (GeneRegionTrack-class), 26
- gene<-, GeneRegionTrack, character-method (GeneRegionTrack-class), 26
- geneDetails (datasets), 20
- geneModels (datasets), 20
- GeneRegionTrack, 36, 95
- GeneRegionTrack (GeneRegionTrack-class), 26
- GeneRegionTrack-class, 26
- genome, RangeTrack-method (RangeTrack-class), 50
- genome, SequenceTrack-method (SequenceTrack-class), 58
- genome<-, IdeogramTrack-method (IdeogramTrack-class), 41
- genome<-, RangeTrack-method (RangeTrack-class), 50
- GenomeAxisTrack, 95
- getPar (DisplayPars-class), 22
- getPar, DisplayPars, character-method (DisplayPars-class), 22
- getPar, DisplayPars, missing-method (DisplayPars-class), 22
- getScheme (settings), 64
- GRanges, 8, 11, 15, 19, 31–33, 40, 44, 46, 55, 63, 92
- group, GeneRegionTrack-method (GeneRegionTrack-class), 26
- group<-, GeneRegionTrack, character-method (GeneRegionTrack-class), 26

- grouping, [9](#), [11](#), [15](#), [19](#), [30](#), [33](#), [36](#), [40](#), [44](#), [46](#), [55](#), [63](#), [75](#), [78](#), [79](#), [82](#), [83](#), [89](#), [93](#)
- Gviz-defunct, [37](#)
- Gviz-deprecated, [37](#)
- HighlightTrack, [9](#), [11](#), [15](#), [19](#), [33](#), [40](#), [44](#), [46](#), [55](#), [63](#), [92](#)
- HighlightTrack (HighlightTrack-class), [37](#)
- HighlightTrack-class, [37](#)
- identifier, GeneRegionTrack-method (GeneRegionTrack-class), [26](#)
- identifier<-, GeneRegionTrack, character-method (GeneRegionTrack-class), [26](#)
- IdeogramTrack (IdeogramTrack-class), [41](#)
- IdeogramTrack-class, [41](#)
- ideoTrack (datasets), [20](#)
- idTrack (datasets), [20](#)
- idxTrack (datasets), [20](#)
- ImageMap, [9](#), [11](#), [15](#), [19](#), [33](#), [40](#), [44](#), [46](#), [48](#), [49](#), [55](#), [63](#), [92](#)
- initialize, AlignmentsTrack-method (AlignmentsTrack-class), [3](#)
- initialize, BiomartGeneRegionTrack-method (BiomartGeneRegionTrack-class), [12](#)
- initialize, CustomTrack-method (CustomTrack-class), [18](#)
- initialize, DetailsAnnotationTrack-method (DetailsAnnotationTrack-class), [20](#)
- initialize, GeneRegionTrack-method (GeneRegionTrack-class), [26](#)
- initialize, HighlightTrack-method (HighlightTrack-class), [37](#)
- initialize, IdeogramTrack-method (IdeogramTrack-class), [41](#)
- initialize, RangeTrack-method (RangeTrack-class), [50](#)
- initialize, ReferenceAlignmentsTrack-method (AlignmentsTrack-class), [3](#)
- initialize, ReferenceGeneRegionTrack-method (GeneRegionTrack-class), [26](#)
- initialize, ReferenceTrack-method (availableDefaultMapping), [10](#)
- initialize, SequenceBSgenomeTrack-method (SequenceBSgenomeTrack-class), [56](#)
- initialize, SequenceDNAStringSetTrack-method (SequenceDNAStringSetTrack-class), [57](#)
- initialize, SequenceRNAStringSetTrack-method (SequenceRNAStringSetTrack-class), [57](#)
- initialize, SequenceTrack-method (SequenceTrack-class), [58](#)
- initialize, StackedTrack-method (StackedTrack-class), [90](#)
- IRanges, [5](#), [9](#), [11](#), [15](#), [19](#), [29](#), [33](#), [38](#), [40](#), [44](#), [46](#), [55](#), [63](#), [92](#)
- iTrack (datasets), [20](#)
- itrack (datasets), [20](#)
- knownGenes (datasets), [20](#)
- lattice::panel.bwplot, [68–71](#), [89](#)
- lattice::panel.grid, [67](#), [68](#), [72–74](#), [77](#), [80](#), [81](#), [84](#), [85](#), [88](#), [89](#)
- lattice::panel.loess, [69](#), [70](#), [89](#)
- lattice::panel.xyplot, [68–70](#), [89](#)
- latticeExtra::panel.horizonplot, [69](#), [70](#), [89](#)
- length, HighlightTrack-method (HighlightTrack-class), [37](#)
- length, IdeogramTrack-method (IdeogramTrack-class), [41](#)
- length, RangeTrack-method (RangeTrack-class), [50](#)
- length, SequenceTrack-method (SequenceTrack-class), [58](#)
- Mart, [13](#), [14](#)
- max, RangeTrack-method (RangeTrack-class), [50](#)
- min, RangeTrack-method (RangeTrack-class), [50](#)
- NumericTrack-class, [45](#)
- panel.grid, [9](#), [12](#), [15](#), [19](#), [33](#), [40](#), [44](#), [46](#), [55](#), [63](#), [93](#)
- plotTracks, [5](#), [9](#), [12](#), [15](#), [19](#), [33](#), [40](#), [44](#), [47](#), [47](#), [55](#), [63](#), [65](#), [66](#), [70](#), [72](#), [73](#), [76](#), [77](#), [79](#), [80](#), [83](#), [85](#), [89](#), [93](#), [95](#)
- position, IdeogramTrack-method (IdeogramTrack-class), [41](#)
- position, RangeTrack-method (RangeTrack-class), [50](#)

- range, RangeTrack-method
(RangeTrack-class), 50
- ranges, RangeTrack-method
(RangeTrack-class), 50
- ranges<-, RangeTrack-method
(RangeTrack-class), 50
- RangeTrack, 9, 11, 15, 19, 33, 40, 44, 46, 55, 63, 92
- RangeTrack-class, 50
- ReferenceTrack-class
(availableDefaultMapping), 10
- refGenes (datasets), 20
- RNASequenceTrack (SequenceTrack-class), 58
- seqinfo, RangeTrack-method
(RangeTrack-class), 50
- seqlevels, RangeTrack-method
(RangeTrack-class), 50
- seqlevels, SequenceBSgenomeTrack-method
(SequenceTrack-class), 58
- seqlevels, SequenceTrack-method
(SequenceTrack-class), 58
- seqnames, RangeTrack-method
(RangeTrack-class), 50
- seqnames, SequenceBSgenomeTrack-method
(SequenceTrack-class), 58
- seqnames, SequenceTrack-method
(SequenceTrack-class), 58
- SequenceBSgenomeTrack-class, 56
- SequenceDNAStringSetTrack-class, 57
- SequenceRNAStringSetTrack-class, 57
- SequenceTrack, 5
- SequenceTrack (SequenceTrack-class), 58
- SequenceTrack-class, 58
- setPar (DisplayPars-class), 22
- setPar, DisplayPars, character-method
(DisplayPars-class), 22
- setPar, DisplayPars, list-method
(DisplayPars-class), 22
- setStacks, AlignmentsTrack-method
(AlignmentsTrack-class), 3
- setStacks, HighlightTrack-method
(HighlightTrack-class), 37
- setStacks, StackedTrack-method
(StackedTrack-class), 90
- settings, 5, 9, 12, 14, 15, 17, 19, 28, 33, 39, 40, 44, 47–49, 55, 60, 63, 64, 93
- show, AlignmentsTrack-method
(AlignmentsTrack-class), 3
- show, CustomTrack-method
(CustomTrack-class), 18
- show, DisplayPars-method
(DisplayPars-class), 22
- show, GeneRegionTrack-method
(GeneRegionTrack-class), 26
- show, HighlightTrack-method
(HighlightTrack-class), 37
- show, IdeogramTrack-method
(IdeogramTrack-class), 41
- show, InferredDisplayPars-method
(DisplayPars-class), 22
- show, ReferenceAlignmentsTrack-method
(AlignmentsTrack-class), 3
- show, ReferenceGeneRegionTrack-method
(GeneRegionTrack-class), 26
- show, ReferenceSequenceTrack-method
(SequenceTrack-class), 58
- show, SequenceBSgenomeTrack-method
(SequenceTrack-class), 58
- show, SequenceDNAStringSetTrack-method
(SequenceTrack-class), 58
- show, SequenceRNAStringSetTrack-method
(SequenceTrack-class), 58
- snpLocations (datasets), 20
- split, RangeTrack, ANY-method
(RangeTrack-class), 50
- StackedTrack, 49
- StackedTrack-class, 90
- stacking, StackedTrack-method
(StackedTrack-class), 90
- stacking<-, StackedTrack, character-method
(StackedTrack-class), 90
- stacks, AlignmentsTrack-method
(AlignmentsTrack-class), 3
- stacks, StackedTrack-method
(StackedTrack-class), 90
- start, IdeogramTrack-method
(IdeogramTrack-class), 41
- start, RangeTrack-method
(RangeTrack-class), 50
- start, SequenceTrack-method
(SequenceTrack-class), 58
- start<-, IdeogramTrack-method
(IdeogramTrack-class), 41
- start<-, RangeTrack-method

(RangeTrack-class), 50
 strand, RangeTrack-method
 (RangeTrack-class), 50
 strand<-, RangeTrack, ANY-method
 (RangeTrack-class), 50
 subseq, ReferenceSequenceTrack-method
 (SequenceTrack-class), 58
 subseq, SequenceTrack-method
 (SequenceTrack-class), 58
 subset, AlignmentsTrack-method
 (AlignmentsTrack-class), 3
 subset, BiomartGeneRegionTrack-method
 (BiomartGeneRegionTrack-class),
 12
 subset, HighlightTrack-method
 (HighlightTrack-class), 37
 subset, RangeTrack-method
 (RangeTrack-class), 50
 subset, ReferenceAlignmentsTrack-method
 (AlignmentsTrack-class), 3
 subset, ReferenceGeneRegionTrack-method
 (GeneRegionTrack-class), 26
 subset, StackedTrack-method
 (StackedTrack-class), 90
 symbol, GeneRegionTrack-method
 (GeneRegionTrack-class), 26
 symbol<-, GeneRegionTrack, character-method
 (GeneRegionTrack-class), 26

 to (datasets), 20
 transcript, GeneRegionTrack-method
 (GeneRegionTrack-class), 26
 transcript<-, GeneRegionTrack, character-method
 (GeneRegionTrack-class), 26
 twoGroups (datasets), 20

 UcscTrack, 93

 values, AlignmentsTrack-method
 (AlignmentsTrack-class), 3
 values, RangeTrack-method
 (RangeTrack-class), 50

 width, IdeogramTrack-method
 (IdeogramTrack-class), 41
 width, RangeTrack-method
 (RangeTrack-class), 50
 width, SequenceTrack-method
 (SequenceTrack-class), 58
 width<-, IdeogramTrack-method
 (IdeogramTrack-class), 41
 width<-, RangeTrack-method
 (RangeTrack-class), 50