Package ‘Gviz’

April 17, 2024

Title Plotting data and annotation information along genomic coordinates

Version 1.47.1

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

RoxygenNote 7.2.3.9001

Roxygen list(markdown = TRUE)

Depends R (>= 4.3), methods, S4Vectors (>= 0.9.25), IRanges (>= 1.99.18), GenomicRanges (>= 1.17.20), grid

Imports XVector (>= 0.5.7), rtracklayer (>= 1.25.13), lattice, RColorBrewer, biomaRt (>= 2.11.0), AnnotationDbi (>= 1.27.5), Biobase (>= 2.15.3), GenomicFeatures (>= 1.17.22), ensembldb (>= 2.11.3), BSgenome (>= 1.33.1), Biostrings (>= 2.33.11), biovizBase (>= 1.13.8), Rsamtools (>= 1.17.28), latticeExtra (>= 0.6-26), matrixStats (>= 0.8.14), GenomicAlignments (>= 1.1.16), GenomeInfoDb (>= 1.1.3), BiocGenerics (>= 0.11.3), digest(>= 0.6.8), graphics, grDevices, stats, utils

Suggests BSgenome.Hsapiens.UCSC.hg19, 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'

git_url https://git.bioconductor.org/packages/Gviz

git_branch devel

git_last_commit d1385ee

git_last_commit_date 2023-11-16

Repository Bioconductor 3.19

Date/Publication 2024-04-16

Author Florian Hahne [aut],
Steffen Durinck [aut],
Robert Ivanek [aut, cre] (<https://orcid.org/0000-0002-8403-056X>),
Arne Mueller [aut],
Steve Lianoglou [aut],
Ge Tan [aut],
Lance Parsons [aut],
Shraddha Pai [aut],
Thomas McCarthy [ctb],
Felix Ernst [ctb],
Mike Smith [ctb]

Maintainer Robert Ivanek <robert.ivanek@unibas.ch>

Contents

AlignmentsTrack-class . . . 3
AnnotationTrack-class . . . 10
availableDefaultMapping . . . 21
BiomartGeneRegionTrack-class . . . 23
collapsing . . . 28
CustomTrack-class . . . 29
datasets . . . 31
DataTrack-class . . . 31
DisplayPars-class . . . 39
exportTracks . . . 43
GdObject-class . . . 44
GeneRegionTrack-class . . . 52

GenomeAxisTrack-class 62
grouping 66
Gviz-defunct 67
Gviz-deprecated 67
HighlightTrack-class 68
IdeogramTrack-class 71
ImageMap-class 74
NumericTrack-class 75
OverlayTrack-class 77
plotTracks 79
RangeTrack-class 83
SequenceTrack-class 88
settings 95
StackedTrack-class 120
UcscTrack 123

Index **127**

`AlignmentsTrack-class` *AlignmentsTrack class and methods*

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'
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 `showMismatchs` 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 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 `character` **string:** the path to a `BAM` file containing the read alignments. To be precise, this will result 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 as well as the optional additional metadata columns `id`, `cigar`, `mapq`, `flag`, `isize`, `groupid`, `status`, `md` and `seqs` (see description of 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 in the separate `chromosome`, `strand`, `feature`, `group` or `id` arguments, because it can not be directly encoded in an `IRanges` object. Note that none of those inputs are mandatory, and if not provided explicitely the more or less reasonable default values `chromosome=NA` and `strand="*"` are used.

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 it will be drawn from the separate `chromosome` or `strand` arguments. In addition, the `id`, `cigar`, `mapq`, `flag`, `isize`, `groupid`, `status`, `md` and `seqs` data can be provided as additional columns. The above comments about potential default values also apply here.

`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 `options(ucscChromosomeNames=TRUE)`. 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 `chrx`, where `x` 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 construct 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.

`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 `NA`.

`stacking`

The stacking type for overlapping items of the track. One in `c(hide, dense, squish, pack, full)`. Currently, only squish (make best use of the available space), dense (no stacking, collapse overlapping ranges), and hide (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 Rsamtools::scanBamFlag(), 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` `DNAStringSet` 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 autodetect 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.
- `ReferenceAlignmentsTrack-class`: The file-based version of the `AlignmentsTrack-class`.
- `initialize(ReferenceAlignmentsTrack)`: Initialize.
- `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 a `AlignmentsTrack` by coordinates and sort if necessary.
- `subset(ReferenceAlignmentsTrack)`: Subset a `ReferenceAlignmentsTrack` by coordinates and sort if necessary.
- `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.hg19)) {
    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
    )
}
```

AnnotationTrack-class *AnnotationTrack class and methods*

Description

A fairly generic track object for arbitrary genomic range annotations, with the option of grouped track items. The extended DetailsAnnotationTrack provides a more flexible interface to add user-defined custom information for each range.

Usage

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

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

AnnotationTrack(
  range = NULL,
  start = NULL,
  end = NULL,
  width = NULL,
  feature,
  group,
  id,
  strand,
  chromosome,
  genome,
  stacking = "squish",
```

```
  name = "AnnotationTrack",
  fun,
  selectFun,
  importFunction,
  stream = FALSE,
  ...
)

DetailsAnnotationTrack(...)

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

## S4 method for signature 'AnnotationTrack'
group(GdObject)

## S4 replacement method for signature 'AnnotationTrack,character'
group(GdObject) <- value

## S4 method for signature 'AnnotationTrack'
identifier(GdObject, type = .dpOrDefault(GdObject, "groupAnnotation", "group"))

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

## S4 method for signature 'AnnotationTrack'
setStacks(GdObject, recomputeRanges = TRUE)

## S4 method for signature 'AnnotationTrack'
consolidateTrack(
  GdObject,
  hasAxis = FALSE,
  hasTitle = .dpOrDefault(GdObject, "showTitle", TRUE),
  title.width = NULL,
  ...
)

## S4 method for signature 'AnnotationTrack'
collapseTrack(GdObject, diff = .pxResolution(coord = "x"), xrange)

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

```
  ...
)

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

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

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

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

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

Arguments

`...` 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 `ReferenceAnnotationTrack` object which will grab the necessary data on the fly during each plotting operation.

`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 `GRanges` object: the genomic ranges for the `Annotation` track as well as the optional additional metadata columns `feature`, `group` and `id` (see description of the individual function parameters below for details). Calling the constructor on a `GRanges` object without further arguments, e.g. `AnnotationTrack(range=obj)` is equivalent to calling the coerce method `as(obj, "AnnotationTrack")`.
- 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 `AnnotationTrack`. 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,"AnnotationTrack")`.
- 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 in the separate `chromosome`, `strand`, `feature`, `group` or `id` arguments, because it can not be directly encoded in an `IRange` object. Note that none of those inputs are mandatory, and if not provided explicitly

the more or less reasonable default values `chromosome=NA` and `strand="*"` are used.

- 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 it will be drawn from the separate `chromosome` or `strand` arguments. In addition, the `feature`, `group` and `id` data can be provided as additional columns. 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 `importFunction` documentation below for further details.

`start`, `end`, `width`

Integer vectors, giving the start and the end 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.

`feature`

Factor (or other vector that can be coerced into one), giving the feature types for the individual track items. When plotting the track to the device, if a display parameter with the same name as the value of `feature` is set, this will be used as the track item's fill colour. See `grouping` for details. Needs to be of equal length as the provided genomic coordinates, or of length 1.

`group`

Factor (or other vector that can be coerced into one), giving the group memberships for the individual track items. When plotting to the device, all items in the same group will be connected. See `grouping` for details. Needs to be of equal length as the provided genomic coordinates, or of length 1.

`id`

Character vector of track item identifiers. When plotting to the device, it's value will be used as the identifier tag if the display parameter `showFeatureId=TRUE`. Needs to be of equal length as the provided genomic ranges, or of length 1.

`strand`

Character vector, the strand information for the individual track items. 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 grouped items need to be on the same strand, 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 `options(ucscChromosomeNames=TRUE)`. 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 `chrx`, where `x` 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 construct 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.

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 NA.

stacking
: The stacking type for overlapping items of the track. One in c(hide, dense, squish, pack,full). Currently, only squish (make best use of the available space), dense (no stacking, collapse overlapping ranges), and hide (do not show any track items at all) are implemented.

name
: Character scalar of the track's name used in the title panel when plotting.

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 explicitely 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.

selectFun
: A function that is being called for each entry in the AnnotationTrack object with exactly the same arguments as in fun. 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 TRUE, details will be drawn for the item, if it is FALSE, the details strip for the item is omitted.

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, bed, bam.

Value

The return value of the constructor function is a new object of class AnnotationTrack or of class DetailsAnnotationTrack, 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(AnnotationTrack): Show method.
- ReferenceAnnotationTrack-class: The file-based version of the AnnotationTrack-class.
- initialize(ReferenceAnnotationTrack): Initialize.
- AnnotationTrack(): Constructor function for AnnotationTrack-class
- DetailsAnnotationTrack-class: directly extends AnnotationTrack.
- 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 location. 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.
- `group(AnnotationTrack)`: extract the group membership for all track items.
- `group(GdObject = AnnotationTrack) <- 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(AnnotationTrack)`: return track item identifiers. Depending on the setting of the optional argument lowest, these are either the group identifiers or the individual item identifiers.
- `identifier(GdObject = AnnotationTrack) <- 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<-`.
- `setStacks(AnnotationTrack)`: Recompute the stacks based on the available space and on the object's track items and stacking settings.
- `consolidateTrack(AnnotationTrack)`: Consolidate. Determine whether there is group label annotation or not, and add this information as the internal display parameter `.__hasAnno`. Precompute the grouped ranges together with optional labels in order to determine the correct plotting range later.
- `collapseTrack(AnnotationTrack)`: preprocess the track before plotting. This will collapse overlapping track items based on the available resolution and increase the width and height of all track objects to a minimum value to avoid rendering issues. See collapsing for details.
- `subset(AnnotationTrack)`: subset a `AnnotationTrack` by coordinates and sort if necessary.
- `subset(ReferenceAnnotationTrack)`: subset a `ReferenceAnnotationTrack` by coordinates and sort if necessary.
- `drawGD(AnnotationTrack)`: 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.
- `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.
- `show(AnnotationTrack)`: Show method.
- `show(ReferenceAnnotationTrack)`: Show method.

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 not 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 name 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 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.

`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.

Objects from the class

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

Author(s)

Florian Hahne, Arne Mueller

See Also

`DisplayPars`

`GdObject`

`GRanges`

`HighlightTrack`

ImageMap

IRanges

RangeTrack

DataTrack

collapsing

grouping

panel.grid

plotTracks

settings

Examples

```
## 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"
)

## Or from a data.frame
df <- data.frame(
    start = st, end = ed, strand = str, id = paste("annTrack item", 1:4),
    feature = "test", group = gr
)
annTrack <- AnnotationTrack(
    range = df, genome = "hg19", chromosome = 7,
    name = "generic annotation", stacking = "squish"
)

## Or from a GRanges object
gr <- GRanges(
    seqnames = "chr7", range = IRanges(start = df$start, end = df$end),
    strand = str
)
genome(gr) <- "hg19"
mcols(gr) <- df[, -(1:3)]
annTrack <- AnnotationTrack(
    range = gr, name = "generic annotation",
    stacking = "squish"
)
```

```
## Finally from a GRangesList
grl <- split(gr, values(gr)$group)
AnnotationTrack(grl)


## Plotting
plotTracks(annTrack)

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

## Subsetting and splitting
subTrack <- subset(annTrack, to = 2155000)
length(subTrack)
subTrack[1:2]
split(annTrack, c(1, 2, 1, 2))

## Accessors
start(annTrack)
end(annTrack)
width(annTrack)
position(annTrack)
width(subTrack) <- width(subTrack) + 1000

strand(annTrack)
strand(subTrack) <- "-"

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

genome(annTrack)
genome(subTrack) <- "mm9"

range(annTrack)
ranges(annTrack)

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

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

values(annTrack)

## Grouping
group(annTrack)
group(subTrack) <- "Group 1"
chromosome(subTrack) <- "chr7"
```

```
plotTracks(subTrack)

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

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

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

## DetailsAnnotationTrack
library(lattice) # need to use grid grapics

## generate two random distributions per row (probe/feature)
## the difference between the distributions increases from probe 1 to 4
m <- matrix(c(rgamma(400, 1)), ncol = 100)
m[, 51:100] <- m[, 51:100] + 0:3
## rownames must be accessible by AnnotationTrack element identifier
rownames(m) <- identifier(annTrack, "lowest")

## create a lattice density plot for the values (signals) of the two groups
## as the chart must be placed into a pre-set grid view port we have to use
## print without calling plot.new! Note, use a common prefix for all lattice.
## Avoid wasting space by removing y-axis decorations.

## Note, in this example 'm' will be found in the environment the 'details'
## function is defined in. To avoid overwriting 'm' you should use a closure
## or environment to access 'm'.
details <- function(identifier, ...) {
    d <- data.frame(signal = m[identifier, ], group = rep(c("grp1", "grp2"), each = 50))
    print(densityplot(~signal,
        group = group, data = d, main = identifier,
        scales = list(draw = FALSE, x = list(draw = TRUE)), ylab = "", xlab = "",
    ), newpage = FALSE, prefix = "plot")
}

deTrack <- AnnotationTrack(
    range = gr, genome = "hg19", chromosome = 7,
    name = "generic annotation with details per entry", stacking = "squish",
    fun = details, details.ratio = 1
)

plotTracks(deTrack)

set.seed(1234)
```

```
deTrack <- AnnotationTrack(
    range = gr, genome = "hg19", chromosome = 7,
    name = "generic annotation with details per entry",
    stacking = "squish", fun = details,
    details.ratio = 1, selectFun = function(...) {
        sample(c(FALSE, TRUE), 1)
    }
)

plotTracks(deTrack)
```

---

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 erence*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.

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 `elemenMetadata` 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 `chrx`, where `x` may be any possible string. The user has to make sure that the respective chromosome is indeed defined for the 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 `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.
- `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 http://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:
bmTrack <- BiomartGeneRegionTrack(
    start = 26682683, end = 26711643,
    chromosome = 7, genome = "mm9"
)

## 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.heigh`) 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.
- `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 *Data sets*

Description

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

DataTrack-class *DataTrack class and methods*

Description

A class to store numeric data values along genomic coordinates. Multiple samples as well as sample groupings are supported, with the restriction of equal genomic coordinates for a single observation across samples.

Usage

```
## S4 method for signature 'DataTrack'
initialize(.Object, data = matrix(), strand, ...)

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

DataTrack(
  range = NULL,
  start = NULL,
  end = NULL,
  width = NULL,
  data,
  chromosome,
```

```
  strand,
  genome,
  name = "DataTrack",
  importFunction,
  stream = FALSE,
  ...
)

## S4 method for signature 'DataTrack'
values(x, all = FALSE)

## S4 replacement method for signature 'DataTrack'
values(x) <- value

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

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

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

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

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

## S4 method for signature 'DataTrack'
collapseTrack(GdObject, diff = .pxResolution(coord = "x"), xrange)

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

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

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

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

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

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

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

Arguments

`.Object` .Object

`data` A numeric matrix of data points with the number of columns equal to the number of coordinates in `range`, or a numeric vector of appropriate length that will be coerced into such a one-row matrix. Each individual row is supposed to contain data for a given sample, where the coordinates for each single observation are constant across samples. Depending on the plotting type of the data (see 'Details' and 'Display Parameters' sections), sample grouping or data aggregation may be available. Alternatively, this can be a character vector of column names that point into the element metadata of the `range` object for subsetting. Naturally, this is only supported when the `range` argument is of class `GRanges`.

`strand` Character vector, the strand information for the individual track items. Currently this has to be unique for the whole track and doesn't really have any visible consequences, but we might decide to make `DataTracks` strand-specific at a later stage.

`...` Additional items which will all be interpreted as further display parameters.

`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 `ReferenceDataTrack` object which will grab the necessary data on the fly during each plotting operation.

`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 `GRanges` **object:** essentially all the necessary information to create a `DataTrack` can be contained in a single `GRanges` object. The track's coordinates are taken from the `start`, `end` and `seqnames` slots, the genome information from the genome slot, and the numeric data values can be extracted from additional metadata columns columns (please note that non-numeric columns are being ignored with a warning). As a matter of fact, calling the constructor on a `GRanges` object without further arguments, e.g. `DataTrack(range=obj)`

is equivalent to calling the coerce method `as(obj, "DataTrack")`. Alternatively, the `GRanges` object may only contain the coordinate information, in which case the numeric data part is expected to be present in the separate `data` argument, and the ranges have to match the dimensions of the data matrix. If `data` is not `NULL`, this will always take precedence over anything defined in the `range` argument. See below for details.

An `IRanges` object: this is very similar to the above case, except that the numeric data part now always has to be provided in the separate `data` argument. Also the chromosome information must be provided in the `chromosome` argument, because neither of the two can be directly encoded in an `IRange` object.

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` column with the chromosome information for each range. If missing it will be drawn from the separate `chromosome` argument. All additional numeric columns will be interpreted as data columns, unless the `data` argument is explicitely provided.

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 `importFunction` documentation below for further details.

`start`, `end`, `width`

Integer vectors, giving the start and the end 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 the single value will be recycled accordingly. Otherwise, the usual R recycling rules for vectors do not apply and the function will cast an error.

`chromosome`

The chromosome on which the track's genomic ranges are defined. A valid UCSC chromosome identifier if `options(ucscChromosomeNames=TRUE)`. 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 `chrx`, where `x` 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 construct 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.

`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 `NA`.

`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 `file` containing the file

path and has to return a proper GRanges object with the data part attached as numeric metadata columns. Essentially the process is equivalent to constructing a DataTrack directly from a GRanges object in that non-numeric columns will be dropped, and further subsetting can be archived by means of the data argument. 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: wig, bigWig/bw, bedGraph and bam.

Some file types support indexing by genomic coordinates (e.g., bigWig and bam), 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 ReferenceDataTrack 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 GRanges object.

value — Value to be set.

GdObject — Object of GdObject-class.

Details

Depending on the setting of the type display parameter, the data can be plotted in various different forms as well as combinations thereof. Supported plotting types are:

- **p:** simple xy-plot.
- **l:** lines plot. In the case of multiple samples this plotting type is not overly usefull since the points in the data matrix are connected in column-wise order. Type a might be more appropriate in these situations.
- **b:** combination of xy-plot and lines plot.
- **a:** lines plot of the column-wise average values.
- **s:** sort and connect data points along the x-axis
- **S:** sort and connect data points along the y-axis
- **g:** add grid lines. To ensure a consitant look and feel across multiple tracks, grid lines should preferentially be added by using the grid display parameter.
- **r:** add a regression line to the plot.
- **h:** histogram-like vertical lines centered in the middle of the coordinate ranges.
- **smooth:** add a loess fit to the plot. The following display parameters can be used to control the loess calculation: span, degree,family, evaluation. See panel.loess for details.
- **histogram:** plot data as a histogram, where the width of the histogram bars reflects the width of the genomic ranges in the range slot.

`mountain`: plot a smoothed version of the data relative to a baseline, as defined by the `baseline` display parameter. The following display parameters can be used to control the smoothing: `span`, `degree`,`family`, `evaluation`. See `panel.loess` for details. The layout of the plot can be further customized via the following display parameters: `col.mountain`, `lwd.mountain`, `lty.mountain`, `fill.mountain`.

`polygon`: plot data as a polygon (similar to `mountain`-type but without smoothing). Data are plotted relative to a baseline, as defined by the `baseline` display parameter. The layout of the plot can be further customized via the following display parameters: `col.mountain`, `lwd.mountain`, `lty.mountain`, `fill.mountain`.

`boxplot`: plot the data as box-and-whisker plots. The layout of the plot can be further customized via the following display parameters: `box.ratio`, `box.width`, `varwidt`, `notch`, `notch.frac`, `levels.fos`, `stats`,`coef`, `do.out`. See `panel.bwplot` for details.

`gradient`: collapse the data across samples and plot this average value as a color-coded gradient. Essenitally this is similar to the heatmap-type plot of a single sample. The layout of the plot can be further customized via the display parameters `ncolor` and `gradient` which control the number of gradient colors as well as the gradient base colors, respectively.

`heatmap`: plot the color-coded values for all samples in the form of a heatmap. The data for individual samples can be visually separated by setting the `separator` display parameter. It's value is taken as the amount of spacing in pixels in between two heatmap rows. The layout of the plot can be further customized via the display parameters `ncolor` and `gradient` which control the number of gradient colors as well as the gradient base colors, respectively.

`horizon`: plot continuous data by cutting the y range into segments and overplotting them with color representing the magnitude and direction of deviation. This is particularly useful when comparing multiple samples, in which case the horizon strips are stacked. See `horizonplot` for details. Please note that the `origin` and `horizonscale` arguments of the Lattice `horizonplot` function are available as display parameters `horizon.origin` and `horizon.scale`.

For some of the above plotting-types the `groups` display parameter can be used to indicate sample sub-groupings. Its value is supposed to be a factor vector of similar length as the number of samples. In most cases, the groups are shown in different plotting colors and data aggregation operations are done in a stratified fashion.

The `window` display parameter can be used to aggregate the data prior to plotting. Its value is taken as the number of equal-sized windows along the genomic coordinates of the track for which to compute average values. The special value `auto` can be used to automatically determine a reasonable number of windows which can be particularly useful when plotting very large genomic regions with many data points.

The `aggregation` parameter can be set to define the aggregation function to be used when averaging in windows or across collapsed items. It takes the form of either a function which should condense a numeric vector into a single number, or one of the predefined options as character scalars `"mean"`, `"median"` or `"sum"` for mean, median or summation, respectively. Defaults to computing mean values for each sample. Note that the predefined options can be much faster because they are optimized to work on large numeric tables.

Value

The return value of the constructor function is a new object of class `DataTrack` or `ReferenceDataTrack`.

Functions

- `initialize(DataTrack)`: Initialize.
- `ReferenceDataTrack-class`: The file-based version of the `DataTrack-class`.
- `initialize(ReferenceDataTrack)`: Initialize.
- `DataTrack()`: Constructor function for `DataTrack-class`
- `values(DataTrack)`: return the raw data values of the object, i.e., the data matrix in the `data` slot.
- `values(DataTrack) <- value`: Replace the data matrix in the `data` slot.
- `strand(DataTrack)`: 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 = DataTrack) <- value`: replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.
- `split(x = DataTrack, f = ANY)`: Split a `DataTrack` object by an appropriate factor vector (or another vector that can be coerced into one). The output of this operation is a list of `DataTrack` objects.
- `feature(DataTrack)`: returns NULL since there is no grouping information for the ranges in a `DataTrack`.
- `feature(GdObject = DataTrack) <- value`: this return the unaltered input object since there is no grouping information for the ranges in a `DataTrack`.
- `collapseTrack(DataTrack)`: preprocess the track before plotting. This will collapse overlapping track items based on the available resolution and increase the width and height of all track objects to a minimum value to avoid rendering issues. See collapsing for details.
- `x[i`: subset the items in the `DataTrack` 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(DataTrack)`: Subset a `DataTrack` by coordinates and sort if necessary.
- `subset(ReferenceDataTrack)`: Subset a `ReferenceDataTrack` by coordinates and sort if necessary.
- `drawAxis(DataTrack)`: add a y-axis to the title panel of a track.
- `drawGD(DataTrack)`: 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(DataTrack)`: Show method.
- `show(ReferenceDataTrack)`: Show method.

Objects from the class

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

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
DataTrack()

## from individual arguments
dat <- matrix(runif(400), nrow = 4)
dtTrack <- DataTrack(
    start = seq(1, 1000, len = 100), width = 10, data = dat,
    chromosome = 1, genome = "mm9", name = "random data"
)

## from GRanges
library(GenomicRanges)
gr <- GRanges(seqnames = "chr1", ranges = IRanges(seq(1, 1000, len = 100),
    width = 10
))
values(gr) <- t(dat)
dtTrack <- DataTrack(range = gr, genome = "mm9", name = "random data")

## from IRanges
dtTrack <- DataTrack(
    range = ranges(gr), data = dat, genome = "mm9",
    name = "random data", chromosome = 1
)
```

```
## from a data.frame
df <- as.data.frame(gr)
colnames(df)[1] <- "chromosome"
dtTrack <- DataTrack(range = df, genome = "mm9", name = "random data")


## Plotting
plotTracks(dtTrack)

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

## Subsetting and splitting
subTrack <- subset(dtTrack, from = 100, to = 300)
length(subTrack)
subTrack[1:2, ]
subTrack[, 1:2]
split(dtTrack, rep(1:2, each = 50))

## Accessors
start(dtTrack)
end(dtTrack)
width(dtTrack)
position(dtTrack)
width(subTrack) <- width(subTrack) - 5

strand(dtTrack)
strand(subTrack) <- "-"

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

genome(dtTrack)
genome(subTrack) <- "mm9"

range(dtTrack)
ranges(dtTrack)

## Data
values(dtTrack)
score(dtTrack)

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

`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, SequenceRNAStringSetTr

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): Alias for the displayPars method.
- getPar(x = DisplayPars, name = missing): Alias for the displayPars method.
- 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")
)
```

GdObject-class *GdObject class and methods*

Description

The virtual parent class for all track items in the Gviz package. This class definition contains all the common entities that are needed for a track to be plotted. During object instantiation for any of the sub-classes inheriting from GdObject, this class' global initializer has to be called in order to assure that all necessary settings are present.

Usage

```
## S4 method for signature 'GdObject'
initialize(.Object, name, ...)

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

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

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

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

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

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

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

## S4 method for signature 'GdObject'
coords(ImageMap)

## S4 method for signature 'GdObject'
tags(ImageMap)

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

## S4 method for signature 'GdObject'
names(x)
```

```
## S4 replacement method for signature 'GdObject,character'
names(x) <- value

group(GdObject, ...)

group(GdObject) <- value

## S4 method for signature 'GdObject'
group(GdObject)

imageMap(GdObject, ...)

## S4 method for signature 'GdObject'
imageMap(GdObject)

imageMap(GdObject) <- value

## S4 replacement method for signature 'GdObject,ImageMapOrNULL'
imageMap(GdObject) <- value

drawAxis(GdObject, ...)

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

drawGrid(GdObject, ...)

drawGD(GdObject, ...)

gene(GdObject, ...)

gene(GdObject) <- value

symbol(GdObject, ...)

symbol(GdObject) <- value

transcript(GdObject, ...)

transcript(GdObject) <- value

exon(GdObject, ...)

exon(GdObject) <- value

feature(GdObject, ...)
```

```
feature(GdObject) <- value

identifier(GdObject, ...)

identifier(GdObject) <- value

chromosome(GdObject, ...)

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

chromosome(GdObject) <- value

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

position(GdObject, ...)

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

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

consolidateTrack(GdObject, ...)

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

stacking(GdObject, ...)

stacking(GdObject) <- value

stacks(GdObject, ...)

setStacks(GdObject, ...)

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

setCoverage(GdObject, ...)
```

Arguments

name	Name of the retrieved parameter.
...	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.
interactive	logical
recursive	logical
asIs	logical
hideInternal	logical
ImageMap	Object of ImageMap-class, containing optional information for an HTML image map.
GdObject	Object of GdObject-class.

Details

Display Parameters:

The following display parameters are set for objects of class GdObject upon instantiation, unless one or more of them have already been set by one of the optional sub-class initializers, which always get precedence over these global defaults. See settings for details on setting graphical parameters for tracks.

- alpha=1 Numeric scalar. The transparency for all track items.
- alpha.title=NULL Numeric scalar. The transparency for the title panel.
- background.legend="transparent" Integer or character scalar. The background colour for the legend.
- background.panel="transparent" Integer or character scalar. The background colour of the content panel.
- background.title="lightgray" Integer or character scalar. The background colour for the title panel.
- cex=1 Numeric scalar. The overall font expansion factor for all text and glyphs, unless a more specific definition exists.
- 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 font size 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="#0080FF" Integer or character scalar. Default line colour setting for all plotting elements, unless there is a more specific control defined elsewhere.
- col.axis="white" Integer or character scalar. The font and line colour for the y axis, if any.
- col.border.title="white" Integer or character scalar. The border colour for the title panels.
- col.frame="lightgray" Integer or character scalar. The line colour used for the panel frame, if frame==TRUE
- col.grid="#808080" Integer or character scalar. Default line colour for grid lines, both when type=="g" in DataTracks and when display parameter grid==TRUE.
- col.line=NULL Integer or character scalar. Default colours for plot lines. Usually the same as the global col parameter.
- col.symbol=NULL Integer or character scalar. Default colours for plot symbols. Usually the same as the global col parameter.

- `col.title="white"` (Aliases `fontcolour.title`) Integer or character scalar. The border colour 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 colour setting for all plotting elements, unless there is a more specific control defined elsewhere.
- `fontcolour="black"` Integer or character scalar. The font colour for all text, unless a more specific definition exists.
- `fontface=1` Integer or character scalar. The font face for all text, unless a more specific definition exists.
- `fontface.title=2` Integer or character scalar. The font face for the title panels.
- `fontfamily="sans"` Integer or character scalar. The font family for all text, unless a more specific definition exists.
- `fontfamily.title="sans"` Integer or character scalar. The font family for the title panels.
- `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 panel.grid for details.
- `lineheight=1` Numeric scalar. The font line height for all text, unless a more specific definition exists.
- `lty="solid"` Numeric scalar. Default line type setting for all plotting elements, unless there is a more specific control defined elsewhere.
- `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=1` Numeric scalar. Default line width 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.grid=1` Numeric scalar. Default line width 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
- `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=0` The rotation angle for all text unless a more specific definition exists.
- `rotation.title=90` (Aliases rotation.title) 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 place holder in case any of the other tracks include a title. The same holds true for axes. Note that the the title panel background colour could be set to transparent in order to completely hide it.
- size=1 Numeric scalar. The relative size of the track. Can be overridden in the plotTracks function.
- v=-1 Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.
- ... additional display parameters are allowed. Those typically take the value of a valid R colour descriptors. The parameter names will later be matched to optional track item types as defined in the 'feature' range attribute, and all tracks of the matched types are coloured accordingly. See the documentation of the GeneRegionTrack and AnnotationTrack classes as well as grouping for details.

Value

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

Functions

- initialize(GdObject): Initialize the object. This involves setting up a new environment for the display parameters and filling it up with the current settings. All arguments that have not been clobbered up by one of the sub-class initializers are considered to be additional display parameters and are also added to the environment. See settings for details on setting graphical parameters for tracks.
- setPar(x = GdObject, value = character): set the single display parameter name to value. Note that display parameters in the GdObject-class are pass-by-reference, so no re-assignment to the symbol obj is necessary. See settings for details on display parameters and customization.
- setPar(x = GdObject, value = list): set display parameters by the values of the named list in value. Note that display parameters in the GdObject-class are pass-by-reference, so no re-assignment to the symbol obj is necessary. See settings for details on display parameters and customization.
- displayPars(x = GdObject) <- value: set display parameters using the values of the named list in value. See settings for details on display parameters and customization.
- getPar(x = GdObject, name = character): alias for the displayPars method. See settings for details on display parameters and customization.
- getPar(x = GdObject, name = missing): alias for the displayPars method. See settings for details on display parameters and customization.
- displayPars(x = GdObject, name = character): list the value of the display parameter name. See settings for details on display parameters and customization.
- displayPars(x = GdObject, name = missing): list the value of all available display parameters. See settings for details on display parameters and customization.

- coords(GdObject): return the coordinates from the internal image map.
- tags(GdObject): return the tags from the internal image map.
- subset(GdObject): subset a GdObject by coordinates. Most of the respective sub-classes inheriting from GdObject overwrite this method, the default is to return the unaltered input object.
- names(GdObject): return the value of the name slot.
- names(x = GdObject) <- value: set the value of the name slot.
- group(): Generics for group.
- group(GdObject) <- value: Generics for group<-.
- group(GdObject): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usually returns NULL.
- imageMap(): Generics for imageMap.
- imageMap(GdObject): Extract the content of the imageMap slot.
- imageMap(GdObject) <- value: Generics for imageMap<-.
- imageMap(GdObject = GdObject) <- value: Replace the content of the imageMap slot.
- drawAxis(): Generics for drawAxis.
- drawAxis(GdObject): add a y-axis to the title panel of a track if necessary. Unless overwritten in one of the sub-classes this usually does not plot anything and returns NULL.
- drawGrid(): Generics for drawGrid.
- drawGD(): Generics for drawGD.
- gene(): Generics for gene.
- gene(GdObject) <- value: Generics for gene<-.
- symbol(): Generics for symbol.
- symbol(GdObject) <- value: Generics for symbol<-.
- transcript(): Generics for transcript.
- transcript(GdObject) <- value: Generics for transcript<-.
- exon(): Generics for exon.
- exon(GdObject) <- value: Generics for exon<-.
- feature(): Generics for feature.
- feature(GdObject) <- value: Generics for feature<-.
- identifier(): Generics for identifier.
- identifier(GdObject) <- value: Generics for identifier<-.
- chromosome(): Generics for chromosome.
- chromosome(GdObject): return the chromosome for which the track is defined.
- chromosome(GdObject) <- value: Generics for chromosome.
- chromosome(GdObject) <- 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.
- position(): Generics for position.

- genome(GdObject): return the track's genome.
- genome(GdObject) <- value: set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.
- consolidateTrack(): Generics for consolidateTrack.
- consolidateTrack(GdObject): Consolidate. Determine whether there is alpha settings or not, and add this information as the internal display parameter .__hasAlphaSupport.
- stacking(): Generics for stacking.
- stacking(GdObject) <- value: Generics for stacking<-.
- stacks(): Generics for stacks.
- setStacks(): Generics for ".
- setStacks(GdObject): set stacks.
- setCoverage(): Generics for ".

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.

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)
```

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(GdObject)
```

```
## S4 replacement method for signature 'GeneRegionTrack,character'
group(GdObject) <- 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 `rstarts` and `rends` for the currently active chromosome. When building a `GeneRegionTrack` from a `TxDb` 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 `ReferenceGeneRegionTrack` 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 consistancy, 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 explicitely 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-readble symbols renders a lot of the class' functionality useles.

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 importFunction documentation below for further details.

`rstarts` 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 approriate number of items in all annotation and coordinates arguments.

`rends` An integer vector of the end coordinates for the actual gene model items. Both `rstarts` and `rends` have to be of equal length.

`rwidths` An integer vector of widths for the actual gene model items. This can be used instead of either `rstarts` or `rends` to specify the range coordinates.

`strand` Character vector, the strand information for the individual track exons. It may be provided in the form `+` for the Watson strand, `-` for the Crick strand or `*` 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.

`feature` 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 `feature` is set, this will be used as the track item's fill color. Additionally, the feature type defines whether an element in the `GeneRegionTrack` is considered to be coding or non-coding. The details section as well as the section about the `thinBoxFeature` display parameter further below has more information on this. See also `grouping` for details.

`exon` Character vector of exon identifiers. It's values will be used as the identifier tag when plotting to the device if the display parameter `showExonId=TRUE`.

`transcript` 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 `showId=TRUE`, and `geneSymbol=FALSE`.

`gene` Factor (or other vector that can be coerced into one), giving the gene memberships for the individual track exons.

`symbol` A factor with human-readable gene name aliases which will be used as labels when `showId=TRUE`, and `geneSymbol=TRUE`.

`chromosome` The chromosome on which the track's genomic ranges are defined. A valid UCSC chromosome identifier if `options(ucscChromosomeNames=TRUE)`. 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 `chrx`, where `x` 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.

`type` type

`x` A valid track object class name, or the object itself, in which case the class is derived directly from it.

`object` object

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 manully construct objects from local data. Connections to particular online data sources should be implemented as sub-classes, and `GeneRegionTrack` is just the commone 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.
- ReferenceGeneRegionTrack-class: The file-based version of the GeneRegionTrack-class.
- initialize(ReferenceGeneRegionTrack): Initialize.
- 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-readble 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(GdObject = 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 = "mm9",
    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 = 26700000, to = 26705000)
length(subTrack)
subTrack <- grTrack[transcript(grTrack) == "ENSMUST00000144140"]
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) <- "hg19"

range(grTrack)
ranges(grTrack)

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

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

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

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

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

transcript(grTrack)
transcript(subTrack) <- c("foo", "bar")
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)
```

GenomeAxisTrack-class *GenomeAxisTrack class and methods*

Description

A class representing a customizable genomic axis.

Usage

```
## S4 method for signature 'GenomeAxisTrack'
initialize(.Object, range, ids, ...)

GenomeAxisTrack(range = NULL, name = "Axis", id, ...)

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

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

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

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

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

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

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

## S4 method for signature 'GenomeAxisTrack'
length(x)
```

```
## S4 method for signature 'GenomeAxisTrack'
values(x)

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

## S4 method for signature 'GenomeAxisTrack'
collapseTrack(
  GdObject,
  min.width = 1,
  min.distance = 0,
  collapse = TRUE,
  diff = .pxResolution(coord = "x"),
  xrange
)

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

## S4 method for signature 'GenomeAxisTrack'
subset(x, from = NULL, to = NULL, sort = FALSE, ...)

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

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

Arguments

`range`	Optional object of class `GRanges` or `IRanges` containing regions to be highlighted on the axis by coloured boxes.
`...`	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 used in the title panel when plotting.
`id`	A character vector of the same length as `range` containing identifiers for the ranges. If missing, the constructor will try to extract the ids from `names(range)`.
`x`	A valid track object class name, or the object itself, in which case the class is derived directly from it.

Details

A `GenomeAxisTrack` can be customized using the familiar display parameters. By providing a `GRanges` or `IRanges` object to the constructor, ranges on the axis can be further highlighted.

With the `scale` display parameter, a small scale indicator can be shown instead of the entire genomic axis. The scale can either be provided as a fraction of the plotting region (it will be rounded

to the nearest human readable absolute value) or as an absolute value and is always displayed in bp, kb, mb or gb units. Note that most display parameters for the `GenomeAxisTrack` are ignored when a scale is used instead of the full axis. In particular, only the parameters `exponent`, `alpha`, `lwd`, `col`, `cex`, `distFromAxis` and `labelPos` are used.

Value

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

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

Functions

- `initialize(GenomeAxisTrack)`: Intialize.
- `GenomeAxisTrack()`: Constructor
- `ranges(GenomeAxisTrack)`: return the genomic coordinates for the track along with all additional annotation information as an object of class `GRanges`.
- `range(GenomeAxisTrack)`: return the genomic coordinates for the track as an object of class `IRanges`. @export
- `start(GenomeAxisTrack)`: return the start coordinates of the track items.
- `start(GenomeAxisTrack) <- value`: replace the start coordinates of the track items.
- `end(GenomeAxisTrack)`: return the end coordinates of the track items.
- `end(GenomeAxisTrack) <- value`: replace the end coordinates of the track items.
- `width(GenomeAxisTrack)`: return the with of the track items in genomic coordinates.
- `length(GenomeAxisTrack)`: return the number of items stored in the ranges slot.
- `values(GenomeAxisTrack)`: return all additional annotation information except for the genomic coordinates for the track items.
- `strand(GenomeAxisTrack)`: 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.
- `collapseTrack(GenomeAxisTrack)`: preprocess the track before plotting. This will collapse overlapping track items based on the available resolution and increase the width and height of all track objects to a minimum value to avoid rendering issues. See collapsing for details.
- `x[i`: subset the items in the `GenomeAxisTrack` 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(GenomeAxisTrack)`: plot subset all the contained tracks in an `GenomeAxisTrack` by coordinates and sort if necessary.
- `drawGD(GenomeAxisTrack)`: 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(GenomeAxisTrack)`: Show method.

Author(s)

Florian Hahne

See Also

DisplayPars

GdObject

GRanges

HighlightTrack

ImageMap

IRanges

RangeTrack

DataTrack

collapsing

grouping

panel.grid

plotTracks

settings

Examples

```
## Construct object
axTrack <- GenomeAxisTrack(
    name = "Axis",
    range <- IRanges(start = c(100, 300, 800), end = c(150, 400, 1000))
)



## Plotting
plotTracks(axTrack, from = 0, to = 1100)

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

## Subsetting and splitting
subTrack <- subset(axTrack, from = 0, to = 500)
length(subTrack)
subTrack[1]
split(axTrack, c(1, 1, 2))

## Accessors
start(axTrack)
end(axTrack)
width(axTrack)
```

```
strand(axTrack)

range(axTrack)
ranges(axTrack)

## Annotation
values(axTrack)

## Grouping
group(axTrack)

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

## adding an axis to another track
data(cyp2b10)
grTrack <- GeneRegionTrack(
    start = 26682683, end = 26711643,
    rstart = cyp2b10$start, rends = cyp2b10$end, chromosome = 7, genome = "mm9",
    transcript = cyp2b10$transcript, gene = cyp2b10$gene, symbol = cyp2b10$symbol,
    name = "Cyp2b10", strand = cyp2b10$strand
)

plotTracks(list(grTrack, GenomeAxisTrack()))
plotTracks(list(grTrack, GenomeAxisTrack(scale = 0.1)))
plotTracks(list(grTrack, GenomeAxisTrack(scale = 5000)))
plotTracks(list(grTrack, GenomeAxisTrack(scale = 0.5, labelPos = "below")))
```

`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 `groups` 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 *Defunct functions in package* Gviz

Description

These functions are defunct and no longer available.

Defunct functions are::

(none)

Gviz-deprecated *Deprecated functions in package* Gviz

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 *HighlightTrack class and methods*

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

`.Object`	.Object
`trackList`	A list of Gviz track objects that all have to inherit from class `GdObject`.
`...`	All additional parameters are ignored.
`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** `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 `start` and `end` 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.
`start, end`	An integer scalar with the genomic start or end coordinates for the highlighting range. Can also be supplied as part of the `range` argument.
`width`	An integer vector of widths for highlighting ranges. This can be used instead of either `start` or `end` to specify the range coordinates.
`chromosome`	The chromosome on which the track's genomic ranges are defined. A valid UCSC chromosome identifier if `options(ucscChromosomeNames=TRUE)`. 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 `chrx`, where `x` 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.
`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`.
`name`	Character scalar of the track's name. This is not really used and only exists fro 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 GdObject-class.
object	object

Details

A track to conceptionally 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.
- 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): Rrecompute 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): Consolidate For a HighlightTrack apply the method on each of the subtracks in the trackList slot
- subset(HighlightTrack): subset all the contained tracks in an 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.

Arguments

chromosome	The chromosome for which to create the ideogram. Has to be a valid UCSC chromosome identifier of the form chrx, 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.
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.
name	Character scalar of the track's name used in the title panel when plotting. Defaults to the selected chromosome.

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.

... — Additional items which will all be interpreted as further display parameters.

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 `IdeogramTracks`, 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-bir`

Value

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

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 acces. 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 = "mm9")

## 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) <- "hg19"

## End(Not run)

range(idTrack)
ranges(idTrack)

## Annotation
values(idTrack)

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

ImageMap-class *ImageMap: HTML image map information*

Description

HTML image map information for annotation tracks.

Usage

```
coords(ImageMap, ...)

## S4 method for signature '`NULL`'
coords(ImageMap)

## S4 method for signature 'ImageMap'
coords(ImageMap)

tags(ImageMap, ...)

## S4 method for signature '`NULL`'
tags(ImageMap)

## S4 method for signature 'ImageMap'
tags(ImageMap)
```

Arguments

ImageMap	Object of ImageMap-class, containing optional information for an HTML image map.

Details

Objects of the ImageMap-class are usually not created by the user, hence the constructor function ImageMap is not exported in the name space.

Value

Returns the coordinates from the image map.

Returns the tags from the image map.

Functions

- coords(): Generics for coords.
- coords(`NULL`): Returns the coordinates from the image map.
- coords(ImageMap): Returns the coordinates from the image map.
- tags(): Generics for tags.

- `tags(`NULL`)`: Returns the tags from the image map
- `tags(ImageMap)`: Returns the tags from the image map

Slots

`coords` Object of class `matrix`, the image map coordinates. In the order x bl, y bl, x tr, y tr. Row names are mandatory for the matrix and have to be unique.

`tags` Object of class `list`, the individual HTML tags for the image map. The value of each list item has to be a named character vector, where the names must match back into the row names of the `coords` matrix

Examples

```
## Not provided. This is an internal structure.
```

`NumericTrack-class` *NumericTrack class and methods*

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 not 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 name 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 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`

OverlayTrack-class *OverlayTrack class and methods*

Description

A container for other track objects from the Gviz package that allows for overlays of their content on the same region of the plot.

Usage

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

OverlayTrack(trackList = list(), name = "OverlayTrack", ...)

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

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

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

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

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

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

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

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

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

Arguments

`trackList`	A list of Gviz track objects that all have to inherit from class `GdObject`.
`...`	All additional parameters are ignored.

name Character scalar of the track's name. This is not really used and only exists fro completeness.

Details

A track to conceptionally group other Gviz track objects into a meta track in order to merge them into a single overlay visualization. Only the first track in the supplied list will be inferred when setting up the track title and axis, for all the other tracks only the panel content is plotted.

Value

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

Functions

- `initialize(OverlayTrack)`: Initialize.
- `OverlayTrack()`: Constructor function for `OverlayTrack-class`.
- `displayPars(x = OverlayTrack) <- value`: set display parameters using the values of the named list in value. See `settings` for details on display parameters and customization.
- `length(OverlayTrack)`: return the number of subtracks.
- `chromosome(OverlayTrack)`: return the chromosome for which the track is defined.
- `chromosome(OverlayTrack) <- 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(OverlayTrack)`: 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(OverlayTrack)`: #' For a `OverlayTrack` apply the method on each of the subtracks in the `trackList` slot
- `subset(OverlayTrack)`: plot subset all the contained tracks in an `OverlayTrack` by coordinates and sort if necessary.
- `drawGD(OverlayTrack)`: 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(OverlayTrack)`: Show method.

Objects from the Class

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

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)
dt1 <- DataTrack(data = dat, start = sort(sample(200, 100)), width = 1, genome = "hg19")
dt2 <- DataTrack(data = dat, start = sort(sample(200, 100)), width = 1, genome = "hg19")
ot <- OverlayTrack(trackList = list(dt1, dt2))
```

---

plotTracks *The main plotting function for one or several Gviz tracks.*

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` Character 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` A 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 plots 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

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(1900000, 2250000, 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 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 parameters
`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 `as.factor(f)` defines the grouping,

Value

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

Functions

- `initialize(RangeTrack)`: Initialize.
- `ranges(RangeTrack)`: return the genomic coordinates for the track along with all additional annotation information as an object of class `GRanges`.
- `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 coordionates, i.e., `(end(obj)-start(obj))/2`.
- `x[i`: 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)`: Consolidate.

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 not 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 name 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 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)
```

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 'SequenceDNAStringSetTrack'
initialize(.Object, sequence, ...)

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

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

## S4 method for signature 'ReferenceSequenceTrack'
initialize(.Object, stream, reference, ...)

## 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'
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:** in this case 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.

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 ReferenceSequenceTrack object which will grab the necessary data on the fly during each plotting operation.

reference — Name of the file (for streatming).

x — A valid track object class name, or the object itself, in which case the class is derived directly from it.

GdObject — the input track object

value — Value to be set.

minBase — Start of the sequence.

maxBase — End of the sequence.

prepare	logical
object	object

Value

The return value of the constructor function is a new object of class SequenceDNAStringSetTrack, SequenceBSgenomeTrack ore 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.
- SequenceTrack(): Constructor
- RNASequenceTrack(): Constructor
- SequenceDNAStringSetTrack-class: The DNAStringSet-based version of the SequenceTrack-class.
- initialize(SequenceDNAStringSetTrack): Initialize.
- SequenceRNAStringSetTrack-class: The RNAStringSet-based version of the SequenceTrack-class.
- initialize(SequenceRNAStringSetTrack): Initialize RNAStringSet-based version of the SequenceTrack-class.
- SequenceBSgenomeTrack-class: The BSgenome-based version of the SequenceTrack-class.
- initialize(SequenceBSgenomeTrack): Initialize.
- ReferenceSequenceTrack-class: The file-based version of the SequenceTrack-class.
- initialize(ReferenceSequenceTrack): Initialize.
- 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 with of the track items in genomic coordinates.
- length(SequenceTrack): return the length of the sequence for active chromosome.
- 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 = "hg19")
sTrack

## Construct from BSGenome object
if (require(BSgenome.Hsapiens.UCSC.hg19)) {
    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) <- "mm9"
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 commom 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 schmeme 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 persitant 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 Givz 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 `DataTrack`s 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 accomodate 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 `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 `DataTrack`s 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 `DataTrack`s 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 definiton 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 `panel.grid` for details.

DataTrack: `aggregateGroups=FALSE`: Logical scalar. Aggregate the values within a sample group using the aggregation funnction 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 intervalls in confint-type plots.

`amount=NULL`: Numeric scalar. Amount of jittering in xy-type plots. See `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 `panel.bwplot` for details.

`box.width=NULL`: Numeric scalar. Parameter controlling the boxplot appearance. See `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 `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.boxplotFrame="#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 `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 `panel.loess` for details.

`do.out=TRUE`: Logical scalar. Parameter controlling the boxplot appearance. See `panel.bwplot` for details.

`evaluation=50`: Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See `panel.loess` for details.

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

`family="symmetric"`: Character scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See `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 `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 `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 `horizonplot` for details.

`jitter.x=FALSE`: Logical scalar. Toggle on jittering on the x axis in xy-type plots. See `panel.xyplot` for details.

`jitter.y=FALSE`: Logical scalar. Toggle off jittering on the y axis in xy-type plots. See `panel.xyplot` for details.

`levels.fos=NULL`: Numeric scalar. Parameter controlling the boxplot appearance. See `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 mimimum 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 `panel.bwplot` for details.

`notch=FALSE`: Logical scalar. Parameter controlling the boxplot appearance. See `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 `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 `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 `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 `DataTrack`s 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 `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 `DataTrack`s 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 `DataTrack`s 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 definiton 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 `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 `DataTrack`s 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 accomodate 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 `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 `DataTrack`s 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 `DataTrack`s 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 definiton 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 `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 `DataTrack`s 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 accomodate 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 `panel.grid` for details.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in `DataTrack`s 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 `DataTrack`s 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 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 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_U
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 `DataTrack`s 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 accomodate 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 `panel.grid` for details.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in `DataTrack`s 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 `DataTrack`s 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 `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_U`
: 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 `DataTrack`s 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 accomodate 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 `panel.grid` for details.

`lty.grid="solid"`: Integer or character scalar. Default line type for grid lines, both when `type=="g"` in `DataTrack`s 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 `DataTrack`s 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 `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. Uselful in combination with a relative setting of `coverageHeight`.

`minSashimiHeight=50`: Integer scalar. The minimum height of the sashimi section. Uselful in combination with a relative setting of `sashimiHeight`.

`noLetters=FALSE`: Logical scalar. Always plot colored boxes for mismatch bases regardles 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 vactor. The type of information to plot. For `coverage` a coverage plot, potentially augmented by base mismatch information, for `sashimi` a sashimi plot, showing the juctions, 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 `DataTrack`s 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 `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 `DataTrack`s 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 `DataTrack`s 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 definiton 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 `panel.grid` for details.

Author(s)

Florian Hahne

See Also

`AnnotationTrack`

`DataTrack`

`DisplayPars`

`GdObject`

`availableDisplayPars`

`collapsing`

`grouping`

`horizonplot`

`panel.bwplot`

`panel.grid`

`panel.loess`

`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 *StackedTrack class and methods*

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.
- `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)`: Consolidate. a display parameter)
- `x[i`: 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 not 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 name 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 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 is 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 codeend="toLoc" to the `UcscTrack` function. The complete function call could look like this:

`UcscTrack(track="foo", genome="mm9", 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-bir`

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 mm9 genome
from <- 65921878
to <- 65980988
knownGenes <- UcscTrack(
    genome = "mm9", 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, 31
∗ **internal**
AnnotationTrack-class, 10
DataTrack-class, 31
GdObject-class, 44
GenomeAxisTrack-class, 62
ImageMap-class, 74
OverlayTrack-class, 77
.DisplayPars *(DisplayPars-class)*, 39
[,DataTrack,ANY,ANY,ANY-method *(DataTrack-class)*, 31
[,GenomeAxisTrack,ANY,ANY,ANY-method *(GenomeAxisTrack-class)*, 62
[,IdeogramTrack,ANY,ANY,ANY-method *(IdeogramTrack-class)*, 71
[,IdeogramTrack,ANY,ANY-method *(IdeogramTrack-class)*, 71
[,IdeogramTrack-method *(IdeogramTrack-class)*, 71
[,RangeTrack,ANY,ANY,ANY-method *(RangeTrack-class)*, 83
[,StackedTrack,ANY,ANY,ANY-method *(StackedTrack-class)*, 120

addScheme *(settings)*, 95
AlignmentsTrack *(AlignmentsTrack-class)*, 3
AlignmentsTrack-class, 3
AnnotationTrack, *28*, *66*, *67*, *95*, *119*, *125*
AnnotationTrack *(AnnotationTrack-class)*, 10
AnnotationTrack-class, 10
as.list,DisplayPars-method *(DisplayPars-class)*, 39
as.list,InferredDisplayPars-method *(DisplayPars-class)*, 39
availableDefaultMapping, 21
availableDisplayPars, *95*, *119*
availableDisplayPars *(DisplayPars-class)*, 39
axTrack *(datasets)*, 31

BiomartGeneRegionTrack, *66*, *67*
BiomartGeneRegionTrack *(BiomartGeneRegionTrack-class)*, 23
BiomartGeneRegionTrack-class, 23
biomTrack *(datasets)*, 31
biomTrack2 *(datasets)*, 31
bmt *(datasets)*, 31
bmTrack *(datasets)*, 31
BSgenome, *91*

chromosome *(GdObject-class)*, 44
chromosome,GdObject-method *(GdObject-class)*, 44
chromosome,OverlayTrack-method *(OverlayTrack-class)*, 77
chromosome,RangeTrack-method *(RangeTrack-class)*, 83
chromosome,SequenceTrack-method *(SequenceTrack-class)*, 88
chromosome<- *(GdObject-class)*, 44
chromosome<-,AlignmentsTrack-method *(AlignmentsTrack-class)*, 3
chromosome<-,GdObject-method *(GdObject-class)*, 44
chromosome<-,HighlightTrack-method *(HighlightTrack-class)*, 68
chromosome<-,IdeogramTrack-method *(IdeogramTrack-class)*, 71
chromosome<-,OverlayTrack-method *(OverlayTrack-class)*, 77
chromosome<-,RangeTrack-method *(RangeTrack-class)*, 83
chromosome<-,SequenceTrack-method *(SequenceTrack-class)*, 88
clearSessionCache, *72*

clearSessionCache (UcscTrack), 123
collapseTrack,AnnotationTrack-method (AnnotationTrack-class), 10
collapseTrack,DataTrack-method (DataTrack-class), 31
collapseTrack,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
collapsing, *9*, *18*, *23*, *26*, 28, *30*, *38*, *51*, *59*, *65*, *71*, *72*, *76*, *79*, *88*, *93*, *97*, *98*, *103–110*, *112–114*, *119*, *123*
conservation (datasets), 31
consolidateTrack (GdObject-class), 44
consolidateTrack,AnnotationTrack-method (AnnotationTrack-class), 10
consolidateTrack,GdObject-method (GdObject-class), 44
consolidateTrack,HighlightTrack-method (HighlightTrack-class), 68
consolidateTrack,OverlayTrack-method (OverlayTrack-class), 77
consolidateTrack,RangeTrack-method (RangeTrack-class), 83
consolidateTrack,SequenceTrack-method (SequenceTrack-class), 88
consolidateTrack,StackedTrack-method (StackedTrack-class), 120
coords (ImageMap-class), 74
coords,GdObject-method (GdObject-class), 44
coords,ImageMap-method (ImageMap-class), 74
coords,NULL-method (ImageMap-class), 74
cpgIslands (datasets), 31
ctrack (datasets), 31
CustomTrack (CustomTrack-class), 29
CustomTrack-class, 29
cyp2b10 (datasets), 31

datasets, 31
DataTrack, *9*, *18*, *23*, *26*, *28*, *30*, *38*, *51*, *59*, *65*, *71*, *72*, *76*, *79*, *87*, *93*, *97–105*, *107*, *110*, *111*, *114*, *115*, *118*, *119*, *123*, *125*
DataTrack (DataTrack-class), 31
DataTrack-class, 31
denseAnnTrack (datasets), 31
DetailsAnnotationTrack (AnnotationTrack-class), 10
DetailsAnnotationTrack-class (AnnotationTrack-class), 10
DisplayPars, *9*, *17*, *22*, *26*, *30*, *38*, *51*, *59*, *65*, *70*, *72*, *76*, *79*, *87*, *93*, *95*, *119*, *122*
DisplayPars (DisplayPars-class), 39
displayPars (DisplayPars-class), 39
displayPars,DisplayPars,character-method (DisplayPars-class), 39
displayPars,DisplayPars,missing-method (DisplayPars-class), 39
displayPars,GdObject,character-method (GdObject-class), 44
displayPars,GdObject,missing-method (GdObject-class), 44
DisplayPars-class, 39
displayPars<- (DisplayPars-class), 39
displayPars<-,DisplayPars,list-method (DisplayPars-class), 39
displayPars<-,GdObject,list-method (GdObject-class), 44
displayPars<-,HighlightTrack,list-method (HighlightTrack-class), 68
displayPars<-,OverlayTrack,list-method (OverlayTrack-class), 77
DNAString, *91*
DNAStringSet, *91*
drawAxis (GdObject-class), 44
drawAxis,AlignmentsTrack-method (AlignmentsTrack-class), 3
drawAxis,DataTrack-method (DataTrack-class), 31
drawAxis,GdObject-method (GdObject-class), 44
drawAxis,NumericTrack-method (NumericTrack-class), 75
drawGD (GdObject-class), 44
drawGD,AlignmentsTrack-method (AlignmentsTrack-class), 3
drawGD,AnnotationTrack-method (AnnotationTrack-class), 10
drawGD,CustomTrack-method (CustomTrack-class), 29
drawGD,DataTrack-method (DataTrack-class), 31
drawGD,DetailsAnnotationTrack-method (AnnotationTrack-class), 10
drawGD,GeneRegionTrack-method (GeneRegionTrack-class), 52

drawGD,GenomeAxisTrack-method
(GenomeAxisTrack-class), 62
drawGD,IdeogramTrack-method
(IdeogramTrack-class), 71
drawGD,OverlayTrack-method
(OverlayTrack-class), 77
drawGD,SequenceTrack-method
(SequenceTrack-class), 88
drawGD,StackedTrack-method
(StackedTrack-class), 120
drawGrid (GdObject-class), 44
drawGrid,NumericTrack-method
(NumericTrack-class), 75
dtHoriz (datasets), 31

end,GenomeAxisTrack-method
(GenomeAxisTrack-class), 62
end,IdeogramTrack-method
(IdeogramTrack-class), 71
end,RangeTrack-method
(RangeTrack-class), 83
end,SequenceTrack-method
(SequenceTrack-class), 88
end<-,GenomeAxisTrack-method
(GenomeAxisTrack-class), 62
end<-,IdeogramTrack-method
(IdeogramTrack-class), 71
end<-,RangeTrack-method
(RangeTrack-class), 83
ensGenes (datasets), 31
exon (GdObject-class), 44
exon,GeneRegionTrack-method
(GeneRegionTrack-class), 52
exon<- (GdObject-class), 44
exon<-,GeneRegionTrack,character-method
(GeneRegionTrack-class), 52
exportTracks, 43

feature (GdObject-class), 44
feature,DataTrack-method
(DataTrack-class), 31
feature,RangeTrack-method
(RangeTrack-class), 83
feature<- (GdObject-class), 44
feature<-,DataTrack,character-method
(DataTrack-class), 31
feature<-,RangeTrack,character-method
(RangeTrack-class), 83
from (datasets), 31

gcContent (datasets), 31
GdObject, *9*, *17*, *22*, *26*, *30*, *38*, *51*, *59*, *65*, *70*, *72*, *76*, *79–81*, *87*, *93*, *95*, *119*, *122*
GdObject-class, 44
gene (GdObject-class), 44
gene,GeneRegionTrack-method
(GeneRegionTrack-class), 52
gene<- (GdObject-class), 44
gene<-,GeneRegionTrack,character-method
(GeneRegionTrack-class), 52
geneDetails (datasets), 31
geneModels (datasets), 31
GeneRegionTrack, *66, 67*, *125*
GeneRegionTrack
(GeneRegionTrack-class), 52
GeneRegionTrack-class, 52
genome,GdObject-method
(GdObject-class), 44
genome,RangeTrack-method
(RangeTrack-class), 83
genome,SequenceTrack-method
(SequenceTrack-class), 88
genome<-,GdObject-method
(GdObject-class), 44
genome<-,IdeogramTrack-method
(IdeogramTrack-class), 71
genome<-,RangeTrack-method
(RangeTrack-class), 83
GenomeAxisTrack, *125*
GenomeAxisTrack
(GenomeAxisTrack-class), 62
GenomeAxisTrack-class, 62
getBM, *25*
getPar (DisplayPars-class), 39
getPar,DisplayPars,character-method
(DisplayPars-class), 39
getPar,DisplayPars,missing-method
(DisplayPars-class), 39
getPar,GdObject,character-method
(GdObject-class), 44
getPar,GdObject,missing-method
(GdObject-class), 44
getScheme (settings), 95
GRanges, *9*, *17*, *22*, *26*, *30*, *38*, *51*, *57–59*, *65*, *70*, *72*, *76*, *79*, *87*, *93*, *122*
group (GdObject-class), 44
group,AnnotationTrack-method
(AnnotationTrack-class), 10

group,GdObject-method (GdObject-class), 44
group,GeneRegionTrack-method (GeneRegionTrack-class), 52
group<- (GdObject-class), 44
group<-,AnnotationTrack,character-method (AnnotationTrack-class), 10
group<-,GeneRegionTrack,character-method (GeneRegionTrack-class), 52
grouping, *9*, *18*, *23*, *26*, *30*, *38*, *51*, *56*, *59*, *65*, 66, *71*, *73*, *76*, *79*, *88*, *93*, *105, 106*, *108, 109*, *112, 113*, *119*, *123*
Gviz-defunct, 67
Gviz-deprecated, 67

HighlightTrack, *9*, *17*, *22*, *26*, *30*, *38*, *51*, *59*, *65*, *71, 72*, *76*, *79*, *87*, *93*, *122*
HighlightTrack (HighlightTrack-class), 68
HighlightTrack-class, 68
horizonplot, *36*, *99, 100*, *119*

identifier (GdObject-class), 44
identifier,AnnotationTrack-method (AnnotationTrack-class), 10
identifier,GeneRegionTrack-method (GeneRegionTrack-class), 52
identifier<- (GdObject-class), 44
identifier<-,AnnotationTrack,character-method (AnnotationTrack-class), 10
identifier<-,GeneRegionTrack,character-method (GeneRegionTrack-class), 52
IdeogramTrack (IdeogramTrack-class), 71
IdeogramTrack-class, 71
ideoTrack (datasets), 31
idTrack (datasets), 31
idxTrack (datasets), 31
ImageMap, *9*, *18*, *22*, *26*, *30*, *38*, *51*, *59*, *65*, *71, 72*, *76*, *79*, *81*, *87*, *93*, *123*
imageMap (GdObject-class), 44
imageMap,GdObject-method (GdObject-class), 44
ImageMap-class, 74
imageMap<- (GdObject-class), 44
imageMap<-,GdObject,ImageMapOrNULL-method (GdObject-class), 44
initialize,AlignmentsTrack-method (AlignmentsTrack-class), 3
initialize,AnnotationTrack-method (AnnotationTrack-class), 10
initialize,BiomartGeneRegionTrack-method (BiomartGeneRegionTrack-class), 23
initialize,CustomTrack-method (CustomTrack-class), 29
initialize,DataTrack-method (DataTrack-class), 31
initialize,DetailsAnnotationTrack-method (AnnotationTrack-class), 10
initialize,GdObject-method (GdObject-class), 44
initialize,GeneRegionTrack-method (GeneRegionTrack-class), 52
initialize,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
initialize,HighlightTrack-method (HighlightTrack-class), 68
initialize,IdeogramTrack-method (IdeogramTrack-class), 71
initialize,OverlayTrack-method (OverlayTrack-class), 77
initialize,RangeTrack-method (RangeTrack-class), 83
initialize,ReferenceAlignmentsTrack-method (AlignmentsTrack-class), 3
initialize,ReferenceAnnotationTrack-method (AnnotationTrack-class), 10
initialize,ReferenceDataTrack-method (DataTrack-class), 31
initialize,ReferenceGeneRegionTrack-method (GeneRegionTrack-class), 52
initialize,ReferenceSequenceTrack-method (SequenceTrack-class), 88
initialize,ReferenceTrack-method (availableDefaultMapping), 21
initialize,SequenceBSgenomeTrack-method (SequenceTrack-class), 88
initialize,SequenceDNAStringSetTrack-method (SequenceTrack-class), 88
initialize,SequenceRNAStringSetTrack-method (SequenceTrack-class), 88
initialize,SequenceTrack-method (SequenceTrack-class), 88
initialize,StackedTrack-method (StackedTrack-class), 120
IRanges, *6*, *9*, *12*, *18*, *22*, *26*, *30*, *34*, *38*, *51*,

55, 59, 65, 69, 71, 72, 76, 79, 87, 93, 123
iTrack (datasets), 31
itrack (datasets), 31

knownGenes (datasets), 31

length,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
length,HighlightTrack-method (HighlightTrack-class), 68
length,IdeogramTrack-method (IdeogramTrack-class), 71
length,OverlayTrack-method (OverlayTrack-class), 77
length,RangeTrack-method (RangeTrack-class), 83
length,SequenceTrack-method (SequenceTrack-class), 88

Mart, 24, 25
max,RangeTrack-method (RangeTrack-class), 83
min,RangeTrack-method (RangeTrack-class), 83

names,GdObject-method (GdObject-class), 44
names<-,GdObject,character-method (GdObject-class), 44
NumericTrack-class, 75

OverlayTrack (OverlayTrack-class), 77
OverlayTrack-class, 77

panel.bwplot, 36, 99–101, 119
panel.grid, 9, 18, 23, 26, 30, 38, 51, 59, 65, 71, 73, 76, 79, 88, 93, 98, 102–105, 107, 108, 110, 111, 115, 118, 119, 123
panel.loess, 35, 36, 99, 101, 119
panel.xyplot, 98–100, 119
plotTracks, 5, 9, 18, 23, 26, 30, 38, 51, 59, 65, 71, 73, 76, 79, 79, 88, 93, 95, 97, 101, 103, 104, 106, 108, 109, 111, 114, 115, 119, 123–125
position (GdObject-class), 44
position,IdeogramTrack-method (IdeogramTrack-class), 71
position,RangeTrack-method (RangeTrack-class), 83

range,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
range,RangeTrack-method (RangeTrack-class), 83
ranges,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
ranges,RangeTrack-method (RangeTrack-class), 83
RangeTrack, 9, 18, 22, 26, 30, 38, 51, 59, 65, 71, 72, 76, 79, 87, 93, 123
RangeTrack-class, 83
ReferenceAlignmentsTrack-class (AlignmentsTrack-class), 3
ReferenceAnnotationTrack-class (AnnotationTrack-class), 10
ReferenceDataTrack-class (DataTrack-class), 31
ReferenceGeneRegionTrack-class (GeneRegionTrack-class), 52
ReferenceSequenceTrack-class (SequenceTrack-class), 88
ReferenceTrack-class (availableDefaultMapping), 21
refGenes (datasets), 31
RNASequenceTrack (SequenceTrack-class), 88

seqinfo,RangeTrack-method (RangeTrack-class), 83
seqlevels,RangeTrack-method (RangeTrack-class), 83
seqlevels,SequenceBSgenomeTrack-method (SequenceTrack-class), 88
seqlevels,SequenceTrack-method (SequenceTrack-class), 88
seqnames,RangeTrack-method (RangeTrack-class), 83
seqnames,SequenceBSgenomeTrack-method (SequenceTrack-class), 88
seqnames,SequenceTrack-method (SequenceTrack-class), 88
SequenceBSgenomeTrack-class (SequenceTrack-class), 88
SequenceDNAStringSetTrack-class (SequenceTrack-class), 88

SequenceRNAStringSetTrack-class (SequenceTrack-class), 88
SequenceTrack, *5*
SequenceTrack (SequenceTrack-class), 88
SequenceTrack-class, 88
setCoverage (GdObject-class), 44
setPar (DisplayPars-class), 39
setPar,DisplayPars,character-method (DisplayPars-class), 39
setPar,DisplayPars,list-method (DisplayPars-class), 39
setPar,GdObject,character-method (GdObject-class), 44
setPar,GdObject,list-method (GdObject-class), 44
setStacks (GdObject-class), 44
setStacks,AlignmentsTrack-method (AlignmentsTrack-class), 3
setStacks,AnnotationTrack-method (AnnotationTrack-class), 10
setStacks,GdObject-method (GdObject-class), 44
setStacks,HighlightTrack-method (HighlightTrack-class), 68
setStacks,OverlayTrack-method (OverlayTrack-class), 77
setStacks,StackedTrack-method (StackedTrack-class), 120
settings, *5*, *9*, *18*, *23*, *25*, *26*, *28*, *30*, *38*, *51*, *54*, *59*, *65*, *70*, *71*, *73*, *76*, *78–81*, *88*, *90*, *93*, 95, *123*
show,AlignmentsTrack-method (AlignmentsTrack-class), 3
show,AnnotationTrack-method (AnnotationTrack-class), 10
show,CustomTrack-method (CustomTrack-class), 29
show,DataTrack-method (DataTrack-class), 31
show,DisplayPars-method (DisplayPars-class), 39
show,GeneRegionTrack-method (GeneRegionTrack-class), 52
show,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
show,HighlightTrack-method (HighlightTrack-class), 68
show,IdeogramTrack-method (IdeogramTrack-class), 71
show,InferredDisplayPars-method (DisplayPars-class), 39
show,OverlayTrack-method (OverlayTrack-class), 77
show,ReferenceAlignmentsTrack-method (AlignmentsTrack-class), 3
show,ReferenceAnnotationTrack-method (AnnotationTrack-class), 10
show,ReferenceDataTrack-method (DataTrack-class), 31
show,ReferenceGeneRegionTrack-method (GeneRegionTrack-class), 52
show,ReferenceSequenceTrack-method (SequenceTrack-class), 88
show,SequenceBSgenomeTrack-method (SequenceTrack-class), 88
show,SequenceDNAStringSetTrack-method (SequenceTrack-class), 88
show,SequenceRNAStringSetTrack-method (SequenceTrack-class), 88
snpLocations (datasets), 31
split,DataTrack,ANY-method (DataTrack-class), 31
split,RangeTrack,ANY-method (RangeTrack-class), 83
StackedTrack, *81*
StackedTrack-class, 120
stacking (GdObject-class), 44
stacking,StackedTrack-method (StackedTrack-class), 120
stacking<- (GdObject-class), 44
stacking<-,StackedTrack,character-method (StackedTrack-class), 120
stacks (GdObject-class), 44
stacks,AlignmentsTrack-method (AlignmentsTrack-class), 3
stacks,StackedTrack-method (StackedTrack-class), 120
start,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
start,IdeogramTrack-method (IdeogramTrack-class), 71
start,RangeTrack-method (RangeTrack-class), 83
start,SequenceTrack-method (SequenceTrack-class), 88
start<-,GenomeAxisTrack-method

(GenomeAxisTrack-class), 62
start<-,IdeogramTrack-method (IdeogramTrack-class), 71
start<-,RangeTrack-method (RangeTrack-class), 83
strand,DataTrack-method (DataTrack-class), 31
strand,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
strand,RangeTrack-method (RangeTrack-class), 83
strand<-,DataTrack,ANY-method (DataTrack-class), 31
strand<-,RangeTrack,ANY-method (RangeTrack-class), 83
subset,AlignmentsTrack-method (AlignmentsTrack-class), 3
subset,AnnotationTrack-method (AnnotationTrack-class), 10
subset,BiomartGeneRegionTrack-method (BiomartGeneRegionTrack-class), 23
subset,DataTrack-method (DataTrack-class), 31
subset,GdObject-method (GdObject-class), 44
subset,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
subset,HighlightTrack-method (HighlightTrack-class), 68
subset,IdeogramTrack-method (IdeogramTrack-class), 71
subset,OverlayTrack-method (OverlayTrack-class), 77
subset,RangeTrack-method (RangeTrack-class), 83
subset,ReferenceAlignmentsTrack-method (AlignmentsTrack-class), 3
subset,ReferenceAnnotationTrack-method (AnnotationTrack-class), 10
subset,ReferenceDataTrack-method (DataTrack-class), 31
subset,ReferenceGeneRegionTrack-method (GeneRegionTrack-class), 52
subset,StackedTrack-method (StackedTrack-class), 120
symbol (GdObject-class), 44
symbol,GeneRegionTrack-method (GeneRegionTrack-class), 52
symbol<- (GdObject-class), 44
symbol<-,GeneRegionTrack,character-method (GeneRegionTrack-class), 52

tags (ImageMap-class), 74
tags,GdObject-method (GdObject-class), 44
tags,ImageMap-method (ImageMap-class), 74
tags,NULL-method (ImageMap-class), 74
to (datasets), 31
transcript (GdObject-class), 44
transcript,GeneRegionTrack-method (GeneRegionTrack-class), 52
transcript<- (GdObject-class), 44
transcript<-,GeneRegionTrack,character-method (GeneRegionTrack-class), 52
twoGroups (datasets), 31

UcscTrack, 123

values,AlignmentsTrack-method (AlignmentsTrack-class), 3
values,DataTrack-method (DataTrack-class), 31
values,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
values,RangeTrack-method (RangeTrack-class), 83
values<-,DataTrack-method (DataTrack-class), 31

width,GenomeAxisTrack-method (GenomeAxisTrack-class), 62
width,IdeogramTrack-method (IdeogramTrack-class), 71
width,RangeTrack-method (RangeTrack-class), 83
width,SequenceTrack-method (SequenceTrack-class), 88
width<-,IdeogramTrack-method (IdeogramTrack-class), 71
width<-,RangeTrack-method (RangeTrack-class), 83