Package 'Gviz'

March 26, 2013

Version 1.2.1

Title Plotting data and annotation information along genomic coordinates

Author Florian Hahne, Steffen Durinck, Robert Ivanek, Arne Mueller, Steve Lianoglou>

Maintainer Florian Hahne <florian.hahne@novartis.com>

Depends R (>= 2.10.0), methods, grid

Imports
IRanges (>= 1.13.19), rtracklayer (>= 1.15.5), lattice,RColorBrewer, biomaRt (>= 2.11.0), Ge-
nomicRanges (>= 1.7.14),AnnotationDbi (>= 1.17.11), Biobase (>= 2.15.3), BiocGenerics
(>= 0.1.4), GenomicFea-
tures (>= 1.9.7), BSgenome (>= 1.25.1),Biostrings (>= 2.25.1), biovizBase (>= 1.5.7)

Suggests xtable, GenomicFeatures, BSgenome.Hsapiens.UCSC.hg19

biocViews Visualization, Microarray

Description Genomic data analyses requires integrated visualization of known genomic informa-
tion and new experimental data. Gviz uses the biomaRt and the rtracklayer packages to per-
form live annotation queries to Ensembl and UCSC and trans-
lates this to e.g. gene/transcript structures in viewports of the grid graphics package. This re-
sults in genomic information plotted together with your data.

Collate Gviz.R AllGenerics.R AllClasses.R Gviz-methods.R

License Artistic-2.0

LazyLoad yes

R topics documented:

AlignedReadTrack-class   *AlignedReadTrack class and methods (NOTE: THIS IS STILL IN DE-*
VELOPMENT AND SUBJECT TO CHANGE)

Description

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

Usage

AlignedReadTrack(range=NULL, start=NULL, end=NULL, width=NULL, chromosome, strand="+", genome
stacking="squish", name="AlignedReadTrack", coverageOnly=FALSE, ...)

Arguments

range   An object of class GRanges, or a data.frame which will be coerced into one in
                 which case it needs to contain at least the three columns:

                 start, end: the start and end coordinates for the track items.
                 strand: the strand information for the track items. It may be provided in the
                     form + for the Watson strand, - for the Crick strand or * for either one of
                     the two.

                 Alternatively, the range argument may be missing, in which case the relevant
                 information has to be provided as individual function arguments (see below).

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 value will be recycled.
                 Otherwise, the usual R recycling rules for vectors do not apply.

strand   Character vector, the strand information for the individual track items. Needs
                 to be of equal length as the start, end or width vectors, 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. Please note that at this stage only syntactic checking takes place, i.e., the argument value needs to be a single integer, numeric character or a character of the form chr$x$, where x may be any possible string. The user has to make sure that the respective chromosome is indeed defined for the the track's genome.
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.
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.
coverageOnly	Instead of storing individual reads, just compute the coverage and store the resulting coverage vector.
...	Additional items which will all be interpreted as further display parameters.

Value

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

Objects from the Class

Objects can be created using the constructor function AlignedReadTrack.

Slots

coverage: Object of class "list", a list of coverage vectors for the plus strand, the minus strand and for both strands combined.

coverageOnly: Object of class "logical", flag to determine whether the object stores read locations or the coverage vectors only.

stacking: Object of class "character", inherited from class StackedTrack

stacks: Object of class "environment", inherited from class StackedTrack

range: Object of class GRanges, inherited from class RangeTrack

chromosome: Object of class "character", inherited from class RangeTrack

genome: Object of class "character", inherited from class RangeTrack

dp: Object of class DisplayPars, inherited from class GdObject

name: Object of class "character", inherited from class GdObject

imageMap: Object of class ImageMap, inherited from class GdObject

Extends

Class "StackedTrack", directly.

Class "RangeTrack", by class "StackedTrack", distance2.

Class "GdObject", by class "StackedTrack", distance3.

Methods

In the following code chunks, obj is considered to be an object of class AlignedReadTrack.

Exported in the name space:

[signature(x="AlignedReadTrack"): subset the items in the AlignedReadTrack. This is essentially similar to subsetting of the GRanges object in the range slot. For most applications, the subset method may be more appropriate. The operation is only supported for objects that still contain all the read locations, i.e., coverageOnly=FALSE.

Additional Arguments:

  i: subsetting indices

Examples:

  obj[1:5]

subset signature(x="AlignedReadTrack"): subset a AlignedReadTrack by coordinates and sort if necessary.

Usage:

subset(x, from, to, sort=FALSE, stacks=FALSE)

Additional Arguments:

  from, to: the coordinates range to subset to.

  sort: sort the object after subsetting. Usually not necessary.

  stacks: recompute the stacking after subsetting which can be expensive and is not always necessary.

Examples:

  subset(obj, from=10, to=20)
  subset(obj, from=10, to=20, sort=TRUE, stacks=FALSE)

split signature(x="AlignedReadTrack"): split an AlignedReadTrack object by an appropriate factor vector (or another vector that can be coerced into one). The output of this operation is a list of AlignedReadTrack objects.

Additional Arguments:

  f: the splitting factor.
  ...: all further arguments are ignored.

Usage:

split(x, f, ...)

Examples:

  split(obj, c("a", "a", "b", "c", "a"))

coverage signature(x="AlignedReadTrack"): return the coverage vector for one of the strands, or the combined vector.

Usage:

coverage(x, strand="*")

Additional Arguments:

  strand: the selector for the strand, + for the Watson strand, - for the Crick strand or * for both strands.

Examples:

  coveraget(obj)
  coverage(obj, strand="-")

Internal methods:

setCoverage signature(GdObject="AlignedReadTrack"): recompute the coverage on the plus
and minus strand as well as for the combined strands and update the respective slot.

Usage:

setCoverage(GdObject)

Examples:

setCoverage(obj)

drawAxis signature(GdObject="AlignedReadTrack"): add a y-axis to the title panel of a track.

Usage:

drawAxis(GdObject, from, to, subset=FALSE, ...)

Additional Arguments:

from, to: compute axis range from the data within a certain coordinates range only.

subset: subset the object prior to calculating the axis ranges. Can be expensive and is not
always needed.

...: all further arguments are ignored.

Examples:

Gviz:::drawAxis(obj)

drawGD signature(gdObject="AlignedReadTrack"): plot the object to a graphics device. The
return value of this method is the input object, potentially updated during the plotting opera-
tion. 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 sub-
setting of the object can be potentially costly, this can be switched off in case subsetting has
already been performed before or is not necessary.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

minBase, maxBase: the coordinate range to plot.

prepare: run method in preparation or in production mode.

subset: subset the object to the visible region or skip the potentially expensive subsetting
operation.

...: all further arguments are ignored.

Examples:

Gviz:::drawGD(obj)

Gviz:::drawGD(obj, minBase=1, maxBase=100)

Gviz:::drawGD(obj, prepare=TRUE,   subset=FALSE)

drawGrid signature(GdObject="AlignedReadTrack"): superpose a grid on top of a track.

Usage:

drawGrid(GdObject, from, to)

Additional Arguments:

from, to: draw grid within a certain coordinates range. This needs to be supplied for the
plotting function to know the current genomic coordinates.

Examples:

Gviz:::drawGrid(obj, from=10, to=100)

initialize signature(.Object="AligendReadTrack"): initialize the object.

show signature(object="AlignedReadTrack"): show a human-readable summary of the object.

Inherited methods:

stacking signature(GdObject="AlignedReadTrack"): return the current stacking type.

> *Usage:*
> stacking(GdObject)
>
> *Examples:*
>
> stacking(obj)

stacking<- signature(GdObject="AlignedReadTrack", value="character"): set the object's stacking type to one in c(hide, dense, squish, pack,full).

> *Usage:*
> stacking<-(GdObject, value)
>
> *Additional Arguments:*
>
> value: replacement value.
>
> *Examples:*
>
> stacking(obj) <- "squish"

setStacks signature(GdObject="AlignedReadTrack"): recompute the stacks based on the available space and on the object's track items and stacking settings.

> *Usage:*
> setStacks(GdObject, from, to)
>
> *Additional Arguments:*
>
> from, to: compute stacking within a certain coordinates range. This needs to be supplied for the plotting function to know the current genomic coordinates.
>
> *Examples:*
>
> Gviz:::setStacks(obj)

stacks signature(GdObject="AlignedReadTrack"): return the stack indices for each track item.

> *Usage:*
> stacks(GdObject)
>
> *Examples:*
>
> Gviz:::stacks(obj)

chromosome signature(GdObject="AlignedReadTrack"): return the chromosome for which the track is defined.

> *Usage:*
> chromosome(GdObject)
>
> *Examples:*
>
> chromosome(obj)

chromosome<- signature(GdObject="AlignedReadTrack"): 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.

> *Usage:*
> chromosome<-(GdObject, value)
>
> *Additional Arguments:*

value: replacement value.

Examples:

chromosome(obj) <- "chr12"

start, end, width signature(x="AlignedReadTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

start(obj)

end(obj)

width(obj)

start<-, end<-, width<- signature(x="AlignedReadTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

start(obj) <- 1:10

end(obj) <- 20:30

width(obj) <- 1

position signature(GdObject="AlignedReadTrack"): the arithmetic mean of the track item's coordionates, i.e., (end(obj)-start(obj))/2.

Usage:

position(GdObject)

Examples:

position(obj)

feature signature(GdObject="AlignedReadTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping or AnnotationTrack and GeneRegionTrack for details.

Usage:

feature(GdObject)

Examples:

feature(obj)

feature<- signature(gdObject="AlignedReadTrack", value="character"): 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 AlignedReadTrack. See grouping or AnnotationTrack and GeneRegionTrack for details.

Usage:

feature<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="AlignedReadTrack"): return the track's genome.

Usage:

genome(x)

Examples:

genome(obj)

genome<- signature(x="AlignedReadTrack"): set the track's genome. Usually this has to be a
valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

genome(obj) <- "mm9"

length signature(x="AlignedReadTrack"): return the number of items in the track.

Usage:

length(x)

Examples:

length(obj)

range signature(x="AlignedReadTrack"): return the genomic coordinates for the track as an
object of class IRanges.

Usage:

range(x)

Examples:

range(obj)

ranges signature(x="AlignedReadTrack"): return the genomic coordinates for the track along
with all additional annotation information as an object of class GRanges.

Usage:

ranges(x)

Examples:

ranges(obj)

strand signature(x="AlignedReadTrack"): 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.

Usage:

strand(x)

Examples:

strand(obj)

strand<- signature(x="AlignedReadTrack"): replace the strand information for the track items.
The replacement value needs to be an appropriate scalar or vector of strand values.

Usage:

strand<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

strand(obj) <- "+"

values signature(x="AlignedReadTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

Usage:

values(x)

Examples:

values(obj)

coerce signature(from="AlignedReadTrack",to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

Examples:

as(obj, "data.frame")

displayPars signature(x="AlignedReadTrack",   name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

displayPars(obj, "col")

displayPars signature(x="AlignedReadTrack",   name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.

Examples:

displayPars(obj)

getPar signature(x="AlignedReadTrack", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

getPar(obj, "col")

getPar signature(x="AlignedReadTrack", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.

Examples:

getPar(obj)

displayPars<- signature(x="AlignedReadTrack",   value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(x="AlignedReadTrack", value="character"): set the single display parameter name to value. Note that display parameters in the AlignedReadTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

name: the name of the display parameter to set.

Examples:

setPar(obj, "col", "red")

setPar signature(x="AlignedReadTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the AlignedReadTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

setPar(obj, list(col="red", lwd=2))

group signature(GdObject="AlignedReadTrack"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

Usage:

group(GdObject)

Examples:

group(obj)

names signature(x="AlignedReadTrack"): return the value of the name slot.

Usage:

names(x)

Examples:

names(obj)

names<- signature(x="AlignedReadTrack", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

names(obj) <- "foo"

coords signature(ImageMap="AlignedReadTrack"): return the coordinates from the internal image map.

Usage:

coords(ImageMap)

Examples:

coords(obj)

tags signature(x="AlignedReadTrack"): return the tags from the internal image map.

Usage:

tags(x)

Examples:

tags(obj)

Display Parameters

The following display parameters are set for objects of class AlignedReadTrack 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.

collapse=FALSE: collapse overlapping ranges and aggregate the underlying data.

detail="coverage": the amount of detail to plot the data. Either coverage to show the coverage only, or reads to show individual reads. For large data sets the latter can be very inefficient. Please note that reads is only available when the object has been created with option coverageOnly=FALSE.

fill="#0080ff": the fill color for the coverage indicator.

size=NULL: the relative size of the track. Defaults to size selection based on the underlying data. Can be overridden in the plotTracks function.

type="histogram": the plot type, one or several in c("p","l", "b", "a", "s", "g", "r", "S", "smooth", "histogram", See the 'Details' section in DataTrack for more information on the individual plotting types.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of AlignedReadTrack objects.

StackedTrack:

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.

GdObject:

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

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 panels.

cex=1: Numeric scalar. The overall font expansion factor for all text.

cex.axis=NULL: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is computed 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="#0080FF": Integer or character scalar. Default line color 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 color for the y axis, if any.

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

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

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

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

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

fontcolor="black": Integer or character scalar. The font color for all text.

fontface=1: Integer or character scalar. The font face for all text.

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.

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

fontsize=12: Numeric scalar. The font size for all text.

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.

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

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

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

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

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

GeneRegionTrack

GRanges

ImageMap

[IRanges](#)

[RangeTrack](#)

[StackedTrack](#)

[collapsing](#)

[grouping](#)

[panel.grid](#)

[plotTracks](#)

[settings](#)

Examples

```
## Construct from individual arguments
arTrack <- AlignedReadTrack(start=runif(1000, 100, 200), width=24,
genome="mm9", chromosome=7, strand=sample(c("+", "-"), 1000, TRUE))


## Plotting
plotTracks(arTrack)

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

## Subsetting and splitting
subTrack <- subset(arTrack, from=110, to=130)
length(subTrack)
subTrack[1:2]
split(arTrack, strand(arTrack))

## Accessors
start(arTrack)
end(arTrack)
width(arTrack)
position(arTrack)
width(subTrack) <- 30

strand(arTrack)
strand(subTrack) <- "-"

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

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

range(arTrack)
ranges(arTrack)

coverage(arTrack)

## Annotation
```

values(arTrack)

Stacking
stacking(arTrack)
stacking(arTrack) <- "dense"

coercion
as(arTrack, "data.frame")

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 flexible interface to add user-defined custom information for each range.

Usage

```
AnnotationTrack(range=NULL, start=NULL, end=NULL, width=NULL, feature, group,
  id, strand="*", chromosome, genome, stacking="squish", name="AnnotationTrack",
  fun, selectFun, ...)
```

Arguments

AnnotationTrack object can be created from a variety of different inputs in order to nicely embed the package into the existing Bioconductor landscape. Since the main components of this class are essentially genomic ranges, the obvious Bioconductor representation is most likely a GRanges object, or, for grouped elements, a GRangesList. However, in certain cases it may be desirable to build the object from individual function arguments.

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:

range   A GRanges object: the genomic ranges for the Annotation track as well as the optional additional elementMetadata 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 consistancy, 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 data 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 only the former two are mandatory (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.

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 color. 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(DetailsAnnotation Typically those identifiers are passed on to the object constructor during instantiation as the id argument.
- index: a counter enumerating the ranges. The AnnotationTrack object is sorted internally for visibility, and the index argument refers to the index of plotting.
- GdObject: a reference to the currently plotted DetailsAnnotationTrack object.
- GdObject.original: a reference to the DetailsAnnotationTrack before any processing like item collapsing has taken place. Essentially, this is the track object as it exists in your working environment.

Additional arguments can be passed to the plotting function by means of the detailsFunArgs argument (see below). Note that the plot must use grid graphics (e.g. function in the 'lattice' package or low-level grid functions). To access a data object such a matrix or data frame within the function you can either store it as a variable in the global environment or, to avoid name space conflicts, you can make it part of the function environment by means of a closure. Alternatively, you may want to explicitly stick it into an environment or pass it along in the detailsFunArgs list. To figure out in your custom plotting function which annotation element is currently being plotted you can either use the identifier which has to be unique for each range element, or you may want to use the genomic position (start/end/strand/chromosome) e.g. if the data is stored in a GRanges object.

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.

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

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.

Objects from the class

Objects can be created using the constructor function AnnotationTrack.

details

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 accoring 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 preassure. 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 separatly 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.

Slots

stacking: Object of class "character", inherited from class StackedTrack

stacks: Object of class "environment", inherited from class StackedTrack

range: Object of class GRanges, inherited from class RangeTrack

chromosome: Object of class "character", inherited from class RangeTrack

genome: Object of class "character", inherited from class RangeTrack

dp: Object of class DisplayPars, inherited from class GdObject

name: Object of class "character", inherited from class GdObject

imageMap: Object of class ImageMap, inherited from class GdObject

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.

Additional display parameters are being inherited from the StackedTrack parent class.

Extends

Class "StackedTrack", directly.

Class "RangeTrack", by class "StackedTrack", distance 2.

Class "GdObject", by class "StackedTrack", distance3.

DetailsAnnotationTrack directly extends AnnotationTrack.

Methods

In the following code chunks, obj is considered to be an object of class AnnotationTrack or DetailsAnnotationTrack.

Exported in the name space:

group signature(GdObject="AnnotationTrack"): extract the group membership for all track items.

Usage:

group(GdObject)

Examples:

   group(obj)

group<- signature(GdObject="AnnotationTrack",   value="character"): 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.

Usage:

group<-(GdObject, value)

Examples:

   group(obj) <- c("a", "a", "b", "c", "a")

identifier signature(GdObject="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.

Usage:

identifier(GdObject, lowest=FALSE)

Additional Arguments:

  lowest: return the lowest-level identifier, i.e., the item IDs, or the higher level group IDs which do not have to be unqiue.

Examples:

   identifier(obj)
   identifier(obj, lowest=TRUE)

identifier<- signature(GdObject="AnnotationTrack",value="character"): 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<-.

Usage:

identifier<-(GdObject, value)

Examples:

   identifier(obj) <- c("foo", "bar")

Internal methods:

coerce signature(from="AnnotationTrack",to="UCSCData"): coerce to a UCSCData object for export to the UCSC genome browser.

Examples:

as(obj, "UCSCData")

collapseTrack signature(GdObject="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.

Usage:

collapseTrack(GdObject, diff=.pxResolution(coord="x"))

Additional Arguments:

diff: the minimum pixel width to display, everything below that will be inflated to a width of diff.

Examples:

Gviz:::collapseTrack(obj)

drawGD signature(GdObject="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.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

minBase, maxBase: the coordinate range to plot.

prepare: run method in preparation or in production mode.

subset: subset the object to the visible region or skip the potentially expensive subsetting operation.

...: all further arguments are ignored.

Examples:

Gviz:::drawGD(obj)

Gviz:::drawGD(obj, minBase=1, maxBase=100)

Gviz:::drawGD(obj, prepare=TRUE,   subset=FALSE)

drawGrid signature(GdObject="AnnotationTrack"): superpose a grid on top of a track.

Usage:

drawGrid(GdObject, from, to)

Additional Arguments:

from, to: integer scalars, draw grid within a certain coordinates range. This needs to be supplied for the plotting function to know the current genomic coordinates.

Examples:

Gviz:::drawGrid(obj, from=10, to=100)

setStacks signature(GdObject="AnnotationTrack"): recompute the stacks based on the available space and on the object's track items and stacking settings.

Usage:

setStacks(GdObject, from, to)

Additional Arguments:

from, to: integer scalars, compute stacking within a certain coordinates range. This needs to be supplied for the plotting function to know the current genomic coordinates.

Examples:

Gviz:::setStacks(obj, from=1, to=100)

initialize signature(.Object="AnnotationTrack"): initialize the object

show signature(object="AnnotationTrack"): show a human-readable summary of the object

Inherited methods:

stacking signature(GdObject="AnnotationTrack"): return the current stacking type.

Usage:

stacking(GdObject)

Examples:

stacking(obj)

stacking<- signature(GdObject="AnnotationTrack", value="character"): set the object's stacking type to one in c(hide, dense, squish, pack,full).

Usage:

stacking<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

stacking(obj) <- "squish"

stacks signature(GdObject="AnnotationTrack"): return the stack indices for each track item.

Usage:

stacks(GdObject)

Examples:

Gviz:::stacks(obj)

[signature(x="AnnotationTrack", i="ANY", j="ANY",drop="ANY"): subset the items in the AnnotationTrack 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.

Additional Arguments:

i, j: subsetting indices, j is ignored.

drop: argument is ignored.

Examples:

obj[1:5]

chromosome signature(GdObject="AnnotationTrack"): return the currently active chromosome for which the track is defined. For consistancy with other Bioconductor packages, the isActiveSeq alias is also provided.

Usage:

chromosome(GdObject)

Examples:

chromosome(obj)

chromosome<- signature(GdObject="AnnotationTrack"): replace the value of the track's active chromosome. This has to be a valid UCSC chromosome identifier or an integer or character scalar that can be reasonably coerced into one, unless options(ucscChromosomeNames=FALSE). For consistancy with other Bioconductor packages, the isActiveSeq<- alias is also provided.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

chromosome(obj) <- "chr12"

start, end, width signature(x="AnnotationTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

start(obj)

end(obj)

width(obj)

start<-, end<-, width<- signature(x="AnnotationTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

start(obj) <- 1:10

end(obj) <- 20:30

width(obj) <- 1

position signature(GdObject="AnnotationTrack"): the arithmetic mean of the track item's co-ordionates, i.e., (end(obj)-start(obj))/2.

Usage:

position(GdObject)

Examples:

position(obj)

feature signature(GdObject="AnnotationTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping for details.

Usage:

feature(GdObject)

Examples:

feature(obj)

feature<- signature(gdObject="AnnotationTrack",   value="character"): 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 AnnotationTrack. See
grouping for details.

Usage:

feature<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="AnnotationTrack"): return the track's genome.

Usage:

genome(x)

Examples:

genome(obj)

genome<- signature(x="AnnotationTrack"): set the track's genome. Usually this has to be a
valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

genome(obj) <- "mm9"

length signature(x="AnnotationTrack"): return the number of items in the track.

Usage:

length(x)

Examples:

length(obj)

range signature(x="AnnotationTrack"): return the genomic coordinates for the track as an ob-
ject of class IRanges.

Usage:

range(x)

Examples:

range(obj)

ranges signature(x="AnnotationTrack"): return the genomic coordinates for the track along
with all additional annotation information as an object of class GRanges.

Usage:

ranges(x)

Examples:

ranges(obj)

split signature(x="AnnotationTrack"): split a AnnotationTrack 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 AnnotationTrack.

Usage:

split(x, f, ...)

Additional Arguments:

  f: the splitting factor.

  ...: all further arguments are ignored.

Examples:

  split(obj, c("a", "a", "b", "c", "a"))

strand signature(x="AnnotationTrack"): 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.

Usage:

strand(x)

Examples:

  strand(obj)

strand<- signature(x="AnnotationTrack"): replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.

Usage:

strand<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

  strand(obj) <- "+"

values signature(x="AnnotationTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

Usage:

values(x)

Examples:

  values(obj)

coerce signature(from="AnnotationTrack",to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

Examples:

  as(obj, "data.frame")

subset signature(x="AnnotationTrack"): subset a AnnotationTrack by coordinates and sort if necessary.

Usage:

subset(x, from, to, sort=FALSE, ...)

Additional Arguments:

  from, to: the coordinates range to subset to.

  sort: sort the object after subsetting. Usually not necessary.

  ...: additional arguments are ignored.

Examples:

   subset(obj, from=10, to=20, sort=TRUE)

displayPars signature(x="AnnotationTrack",   name="character"): list the value of the
   display parameter name. See settings for details on display parameters and customization.

   Usage:

   displayPars(x, name)

   Examples:

     displayPars(obj, "col")

displayPars signature(x="AnnotationTrack",   name="missing"): list the value of all avail-
   able display parameters. See settings for details on display parameters and customization.

   Examples:

     displayPars(obj)

getPar signature(x="AnnotationTrack", name="character"): alias for the displayPars method.
   See settings for details on display parameters and customization.

   Usage:

   getPar(x, name)

   Examples:

     getPar(obj, "col")

getPar signature(x="AnnotationTrack", name="missing"): alias for the displayPars method.
   See settings for details on display parameters and customization.

   Examples:

     getPar(obj)

displayPars<- signature(x="AnnotationTrack",   value="list"): set display parameters
   using the values of the named list in value. See settings for details on display parameters and
   customization.

   Usage:

   displayPars<-(x, value)

   Examples:

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

setPar signature(x="AnnotationTrack", value="character"): set the single display parameter
   name to value. Note that display parameters in the AnnotationTrack class are pass-by-
   reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on
   display parameters and customization.

   Usage:

   setPar(x, name, value)

   Additional Arguments:

     name: the name of the display parameter to set.

   Examples:

     setPar(obj, "col", "red")

setPar signature(x="AnnotationTrack", value="list"): set display parameters by the values of
   the named list in value. Note that display parameters in the AnnotationTrack class are pass-
   by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on
   display parameters and customization.

   Examples:

     setPar(obj, list(col="red", lwd=2))

names signature(x="AnnotationTrack"): return the value of the name slot.

> *Usage:*
>
> names(x)
>
> *Examples:*
>
> names(obj)

names<- signature(x="AnnotationTrack", value="character"): set the value of the name slot.

> *Usage:*
>
> names<-(x, value)
>
> *Examples:*
>
> names(obj) <- "foo"

coords signature(ImageMap="AnnotationTrack"): return the coordinates from the internal image map.

> *Usage:*
>
> coords(ImageMap)
>
> *Examples:*
>
> coords(obj)

tags signature(x="AnnotationTrack"): return the tags from the internal image map.

> *Usage:*
>
> tags(x)
>
> *Examples:*
>
> tags(obj)

Display Parameters

The following display parameters are set for objects of class AnnotationTrack 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.

> cex=1: Numeric scalar. The font expansion factor for item identifiers.
>
> cex.group=0.6: Numeric scalar. The font expansion factor for the group-level annotation.
>
> col="transparent": Character or integer scalar. The border color for all track items.
>
> col.line="darkgray": Character scalar. The color used for connecting lines between grouped items. Defaults to a dark gray, but if set to NULL the same color as for the first item in the group is used.
>
> fill="lightblue": Character or integer scalar. The fill color for untyped items. This is also used to connect grouped items. See grouping for details.
>
> fontcolor="white": Character or integer scalar. The font color for item identifiers.
>
> fontcolor.group="#808080": Character or integer scalar. The font color for the group-level annotation.
>
> fontface=1: Integer scalar. The font face for item identifiers.
>
> fontface.group=2: Numeric scalar. The font face for the group-level annotation.
>
> fontfamily="sans": Character scalar. The font family for item identifiers.
>
> fontsize=12: Numeric scalar. The font size for item identifiers.

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.

rotation=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, 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.

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

DetailsAnnotationTrack adds the following additional display parameters:

details.size=0.5: Numeric scalar. The fraction of vertical space of the track used for the details section.

details.minWidth=100: Numeric scalar. The minium width in pixels for a details panel, if less space is available no details are plotted.

detailsConnector.col="darkgray": Character or integer scalar. Color of the line connecting the AnnotstionTrack item with its details panel.

detailsConnector.lty="dashed": Character or integer scalar. Type of connecting line.

detailsConnector.lwd=1: Integer scalar. Line width of the connector.

detailsConnector.pch=20: Integer scalar. Type of the connector's ends.

detailsConnector.cex=1: Numeric scalar. Relative size of the connector's end points.

detailsBorder.lty="solid": Character or integer scalar. Line type of the border around each details panel.

detailsBorder.lwd=1: Integer scalar. Line width of the border.

detailsBorder.col="darkgray": Character or integer scalar. Line color of the border.

detailsBorder.fill="transparent": Character or integer scalar. Background color of the border.

details.ratio=Inf: Numeric scalar. By default, the plotting method tries to fill all available space of the details panel tiles. Depending on the dimensions of your plot and the number of tiles this may lead to fairly stretched plots. Restricting the ration of width over height can help to fine tune for somewhat more sane graphics in these cases. Essentially this adds some white space in between individual tiles to force the desired ratio. Together with the size and details.size arguments, which control the vertical extension of the whole track and of the details section, this allows for some fairly generic resizing of the tiles.

detailsFunArgs=list(): List. Additional arguments that get passed on the the details plotting function.

groupDetails=FALSE: Logial scalar. Plot details for feature groups rather than for individual features.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of AnnotationTrack DetailsAnnotationTrack objects.

StackedTrack:

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.

GdObject:

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

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 panels.

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

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

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

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

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

col.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 font color for the title panels.

collapse=TRUE: Boolean controlling wether 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.

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

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

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

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, Arne Mueller

See Also

DisplayPars

GdObject

GRanges

ImageMap

IRanges

RangeTrack

StackedTrack

collapsing

DataTrack

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 <- GenomicRanges::GRanges(seqnames="chr7", range=IRanges::IRanges(start=df$start, end=df$end), strand=str)
genome(gr) <- "hg19"
IRanges::elementMetadata(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=TRUE)
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=TRUE)

## 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)
```

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

```
BiomartGeneRegionTrack(start, end, biomart, chromosome, strand="*",
genome=NULL, stacking="squish", filters=list(), name="BiomartGeneRegionTrack", ...)
```

Arguments

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.
strand	Character scalar, the strand for which to fetch gene information from Biomart. One in +, -, or +-.
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.
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.
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.
...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.

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.

Objects from the class

Objects can be created using the constructor function BiomartGeneRegionTrack.

Slots

biomart: Object of class "MartOrNULL", the connection to the Ensembl Biomart webservice.

filter: Object of class "list", additional filters for the data base query.

start: Object of class "numeric", inherited from class GeneRegionTrack. The start coordinates of the annotation range. The coorrdinates for the individual gene model items are stored in the range slot.

end: Object of class "numeric", inherited from class GeneRegionTrack. The end coordinates of the annotation range. The corrdinates for the individual gene model items are stored in the range slot.

stacking: Object of class "character", inherited from class StackedTrack

stacks: Object of class "environment", inherited from class StackedTrack

range: Object of class GRanges, inherited from class RangeTrack

chromosome: Object of class "character", inherited from class RangeTrack

genome: Object of class "character", inherited from class RangeTrack

dp: Object of class DisplayPars, inherited from class GdObject

name: Object of class "character", inherited from class GdObject

imageMap: Object of class ImageMap, inherited from class GdObject

Extends

Class "GeneRegionTrack", directly.

Class "AnnotationTrack", by class "GeneRegionTrack", distance 2.

Class "StackedTrack", by class "GeneRegionTrack", distance 3.

Class "RangeTrack", by class "GeneRegionTrack", distance 4.

Class "GdObject", by class "GeneRegionTrack", distance 5.

Methods

In the following code chunks, obj is considered to be an object of class BiomartGeneRegionTrack.

Internal methods:

initialize signature(.Object = "BiomartGeneRegionTrack"): initialize the object.

Inherited methods:

group signature(gdObject="BiomartGeneRegionTrack"): extract the group membership for all track items.

Usage:

group(GdObject)

Examples:

group(obj)

group<- signature(gdObject="BiomartGeneRegionTrack", value="character"): 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.

Usage:

group<-(GdObject, value)

Examples:

group(obj) <- c("a", "a", "b", "c", "a")

identifier signature(gdObject="BiomartGeneRegionTrack"): return track item identifiers. Depending on the setting of the optional argument lowest, these are either the group identifiers or the individual item identifiers.

Usage:

identifier(GdObject, lowest=FALSE)

Additional Arguments:

lowest: return the lowest-level identifier, i.e., the item IDs, or the higher level group IDs which do not have to be unqiue.

Examples:

identifier(obj, lowest=FALSE)

identifier<- signature(gdObject="BiomartGeneRegionTrack",value="character"): 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<-.

Usage:

identifier<-(GdObject, value)

Examples:

identifier(obj) <- c("foo", "bar")

exon signature(GdObject="BiomartGeneRegionTrack"): Extract the exon identifiers for all exons in the gene models.

Usage:

exon(GdObject)

Examples:

exon(obj)

exon<- signature(GdObject="BiomartGeneRegionTrack",   value="character"): 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.

Usage:

exon<-(GdObject, value)

Examples:

exon(obj) <- paste("Exon", 1:5)

gene signature(GdObject="BiomartGeneRegionTrack"): Extract the gene identifiers for all
gene models.

Usage:

gene(GdObject)

Examples:

gene(obj)

gene<- signature(GdObject="BiomartGeneRegionTrack",value="character"): 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.

Usage:

gene<-(GdObject, value)

Examples:

gene(obj) <- paste("Gene", LETTERS[1:5])

symbol signature(GdObject="BiomartGeneRegionTrack"): Extract the human-readble gene
symbol for all gene models.

Usage:

symbol(GdObject)

Examples:

symbol(obj)

symbol<- signature(GdObject="BiomartGeneRegionTrack",value="character"): 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.

Usage:

gene<-(GdObject, value)

Examples:

symbol(obj) <- letters[1:5]

transcript signature(GdObject="BiomartGeneRegionTrack"): Extract the transcript identi-
fiers for all transcripts in the gene models.

Usage:

transcript(GdObject)

Examples:

transcript(obj)

transcript<- signature(GdObject="BiomartGeneRegionTrack",value="character"): 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.

Usage:

transcript<-(GdObject, value)

Examples:

transcript(obj) <- paste("Exon", 1:5)

Internal methods:

coerce signature(from="BiomartGeneRegionTrack",to="UCSCData"): coerce to a UCSCData object for export to the UCSC genome browser.

Examples:

as(obj, "UCSCData")

collapseTrack signature(GdObject="BiomartGeneRegionTrack"): 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.

Usage:

collapseTrack(GdObject, diff=.pxResolution(coord="x"))

Additional Arguments:

diff: the minimum pixel width to display, everything below that will be inflated to a width of diff.

Examples:

Gviz:::collapseTrack(obj)

show signature(object="BiomartGeneRegionTrack"): show a human-readable summary of the object

drawGD signature(GdObject="BiomartGeneRegionTrack"): 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.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

minBase, maxBase: the coordinate range to plot.

prepare: run method in preparation or in production mode.

subset: subset the object to the visible region or skip the potentially expensive subsetting operation.

...: all further arguments are ignored.

Examples:

Gviz:::drawGD(obj)
Gviz:::drawGD(obj, minBase=1, maxBase=100)
Gviz:::drawGD(obj, prepare=TRUE,   subset=FALSE)

drawGrid signature(GdObject="BiomartGeneRegionTrack"): superpose a grid on top of a track.

Usage:

drawGrid(GdObject, from, to)

Additional Arguments:

from, to: integer scalars, draw grid within a certain coordinates range. This needs to be supplied for the plotting function to know the current genomic coordinates.

Examples:

Gviz:::drawGrid(obj, from=10, to=100)

setStacks signature(GdObject="BiomartGeneRegionTrack"): recompute the stacks based on
the available space and on the object's track items and stacking settings.

Usage:

setStacks(GdObject, from, to)

Additional Arguments:

from, to: integer scalars, compute stacking within a certain coordinates range. This needs to
be supplied for the plotting function to know the current genomic coordinates.

Examples:

Gviz:::setStacks(obj, from=1, to=100)

stacking signature(GdObject="BiomartGeneRegionTrack"): return the current stacking type.

Usage:

stacking(GdObject)

Examples:

stacking(obj)

stacking<- signature(GdObject="BiomartGeneRegionTrack",   value="character"): set
the object's stacking type to one in c(hide, dense, squish, pack,full).

Usage:

stacking<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

stacking(obj) <- "squish"

stacks signature(GdObject="BiomartGeneRegionTrack"): return the stack indices for each
track item.

Usage:

stacks(GdObject)

Examples:

Gviz:::stacks(obj)

[signature(x="BiomartGeneRegionTrack", i="ANY", j="ANY",drop="ANY"): subset the
items in the BiomartGeneRegionTrack 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.

Additional Arguments:

i, j: subsetting indices, j is ignored.

drop: argument is ignored.

Examples:

obj[1:5]

chromosome signature(GdObject="BiomartGeneRegionTrack"): return the chromosome for
which the track is defined.

Usage:

chromosome(GdObject)

Examples:

chromosome(obj)

chromosome<- signature(GdObject="BiomartGeneRegionTrack"): 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.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

  value: replacement value.

Examples:

  chromosome(obj) <- "chr12"

start, end, width signature(x="BiomartGeneRegionTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

  start(obj)

  end(obj)

  width(obj)

start<-, end<-, width<- signature(x="BiomartGeneRegionTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

  start(obj) <- 1:10

  end(obj) <- 20:30

  width(obj) <- 1

position signature(GdObject="BiomartGeneRegionTrack"): the arithmetic mean of the track item's coordionates, i.e., (end(obj)-start(obj))/2.

Usage:

position(GdObject)

Examples:

  position(obj)

feature signature(GdObject="BiomartGeneRegionTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping for details.

Usage:

feature(GdObject)

Examples:

feature(obj)

feature<- signature(gdObject="BiomartGeneRegionTrack", value="character"): 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 BiomartGeneRegionTrack. See grouping for details.

Usage:

feature<-(GdObject, value)

Additional Arguments:

  value: replacement value.

Examples:

  feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="BiomartGeneRegionTrack"): return the track's genome.

Usage:

genome(x)

Examples:

  genome(obj)

genome<- signature(x="BiomartGeneRegionTrack"): set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

  genome(obj) <- "mm9"

length signature(x="BiomartGeneRegionTrack"): return the number of items in the track.

Usage:

length(x)

Examples:

  length(obj)

range signature(x="BiomartGeneRegionTrack"): return the genomic coordinates for the track as an object of class IRanges.

Usage:

range(x)

Examples:

  range(obj)

ranges signature(x="BiomartGeneRegionTrack"): return the genomic coordinates for the track along with all additional annotation information as an object of class GRanges.

Usage:

ranges(x)

Examples:

  ranges(obj)

split signature(x="BiomartGeneRegionTrack"): split a BiomartGeneRegionTrack 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 BiomartGeneRegionTrack.

Usage:

split(x, f, ...)

Additional Arguments:

   f: the splitting factor.

   ...: all further arguments are ignored.

Examples:

   split(obj, c("a", "a", "b", "c", "a"))

strand signature(x="BiomartGeneRegionTrack"): 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.

Usage:

strand(x)

Examples:

   strand(obj)

strand<- signature(x="BiomartGeneRegionTrack"): replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.

Usage:

strand<-(x, value)

Additional Arguments:

   value: replacement value.

Examples:

   strand(obj) <- "+"

values signature(x="BiomartGeneRegionTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

Usage:

values(x)

Examples:

   values(obj)

coerce signature(from="BiomartGeneRegionTrack",to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

Examples:

   as(obj, "data.frame")

subset signature(x="BiomartGeneRegionTrack"): subset a BiomartGeneRegionTrack by coordinates and sort if necessary.

Usage:

subset(x, from, to, sort=FALSE, ...)

Additional Arguments:

   from, to: the coordinates range to subset to.

   sort: sort the object after subsetting. Usually not necessary.

   ...: additional arguments are ignored.

Examples:

subset(obj, from=10, to=20, sort=TRUE)

displayPars signature(x="BiomartGeneRegionTrack", name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

displayPars(obj, "col")

displayPars signature(x="BiomartGeneRegionTrack", name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.

Examples:

displayPars(obj)

getPar signature(x="BiomartGeneRegionTrack", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

getPar(obj, "col")

getPar signature(x="BiomartGeneRegionTrack", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.

Examples:

getPar(obj)

displayPars<- signature(x="BiomartGeneRegionTrack", value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(x="BiomartGeneRegionTrack", value="character"): set the single display parameter name to value. Note that display parameters in the BiomartGeneRegionTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

name: the name of the display parameter to set.

Examples:

setPar(obj, "col", "red")

setPar signature(x="BiomartGeneRegionTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the BiomartGeneRegionTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

> setPar(obj, list(col="red", lwd=2))

names signature(x="BiomartGeneRegionTrack"): return the value of the name slot.

> *Usage:*
>
> names(x)
>
> *Examples:*
>
> names(obj)

names<- signature(x="BiomartGeneRegionTrack", value="character"): set the value of the name slot.

> *Usage:*
>
> names<-(x, value)
>
> *Examples:*
>
> names(obj) <- "foo"

coords signature(ImageMap="BiomartGeneRegionTrack"): return the coordinates from the internal image map.

> *Usage:*
>
> coords(ImageMap)
>
> *Examples:*
>
> coords(obj)

tags signature(x="BiomartGeneRegionTrack"): return the tags from the internal image map.

> *Usage:*
>
> tags(x)
>
> *Examples:*
>
> tags(obj)

Display Parameters

The following display parameters are set for objects of class BiomartGeneRegionTrack 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.

> 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'.
>
> 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'.

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'.

protein_coding="gold4": 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'.

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

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'.

scRNA="darkorange": 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'.

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'.

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'.

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

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

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of BiomartGeneRegionTrack objects.

GeneRegionTrack:

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

geneSymbols=TRUE: Logical scalar. Use human-readable gene symbols or gene IDs for the transcript annotation.

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

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

AnnotationTrack:

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

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

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

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

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

fontface=1: Integer scalar. The font face for item identifiers.

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

fontfamily="sans": Character scalar. The font family for item identifiers.

fontsize=12: Numeric scalar. The font size for item identifiers.

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.

rotation=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.

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

StackedTrack:

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.

GdObject:

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

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 panels.

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

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

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

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

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

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

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

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

collapse=TRUE: Boolean controlling wether 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.

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

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

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

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

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

References

EBI Biomart webservice at http://www.biomart.org.

See Also

AnnotationTrack

DisplayPars

GdObject

GeneRegionTrack

GRanges

ImageMap

IRanges

Mart

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=TRUE)
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)
```

---

bmTrack   *Data sets*

Description

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

---

collapsing   *Dynamic content based on the available resolution*

Description

When plotting features linearily along genomic coordinates one frequently runs into the problem of too little resolution to adequatelty display all details. Most genome browsers try to reasonably reduce the amount of detail that is shown based on the current zoomn 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 overplotting and prevents rendering issues, e.g. when lines are too thin to be plotted. This feature can be toggled on or off using the collapse display parameter (see settings for details on setting these parameters).

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

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

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

DataTrack(range=NULL, start=NULL, end=NULL, width=NULL, data, chromosome, strand="*", genome, name="DataTrack", ...)

Arguments

We tried to keep instantiation of DataTrack objects as flexible as possible to accomodate different use cases. For instance, one natural way to create a DataTrack is from an existing GRanges object. In other cases it might be more appropriate to build the object using individual function arguments.

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:

range   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 elementMetadata 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 explicitly provided.

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.

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.

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.

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

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.mounta

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, See panel.bwplot for details.

gradient: collapse the data across samples and plot this average value as a color-coded gradient. Essentially 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.

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.

Objects from the class

Objects can be created using the constructor function DataTrack.

Slots

data: Object of class "matrix", containing the data values to be plotted. Individual rows of the matrix correspond to individual samples, and the number of columns has to be identical to the feature number of the GRanges object in the range slot.

strand: Object of class "character", the strand information for the track, in the form '+' for the Watson strand, '-' for the Crick strand or '*' for either of the two.

range: Object of class IRanges, inherited from class RangeTrack. The genomic coordinates for the data values. The length of the object needs to be identical to the number of columns of the data matrix in the data slot.

chromosome: Object of class "character", inherited from class RangeTrack

genome: Object of class "character", inherited from class RangeTrack

dp: Object of class DisplayPars, inherited from class GdObject

name**:** Object of class "character", inherited from class GdObject

imageMap**:** Object of class ImageMap, inherited from class GdObject

Extends

Class "NumericTrack", directly.

Class "RangeTrack", by class "NumericTrack", distance 2.

Class "GdObject", by class "NumericTrack", distance 3.

Methods

In the following code chunks, obj is considered to be an object of class DataTrack.

Exported in the name space:

[signature(x="DataTrack"): subsetting of the object, either to a subet of coordinates, or to a subset of samples.

Additional Arguments:

i, j: subsetting indices for coordinates (i) or samples (j).

Examples:

obj[1:3,]
obj[,2:4]

values signature(x="DataTrack"): return the raw data values of the object, i.e., the data matrix in the data slot.

Usage:

values(x)

Examples:

values(obj)

values<- signature(x="DataTrack"): replace the data matrix in the data slot.

Usage:

values<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

values(obj) <- matrix(1:10, ncol=2)

score signature(x="DataTrack"): return processed data values of the object exactly like they would be plotted to the device (modulo any potential aggregration or collapsing), i.e., the raw data with optional transformations applied.

Usage:

score(x, from=NULL, to=NULL, sort=FALSE, transformation=TRUE)

Additional Arguments:

from, to: restrict to data within a certain coordinates range.

sort: sort the return values by coordinates. This is usually not necessary since the data should already be ordererd, however this is not formaly checked anywhere and some operations strictly depend on ordered data.

transformation: apply a data transformation in case one is defined as the transformation display parameter.

Examples:

  score(obj)
  score(obj, from=100, to=10000)
  score(obj, sort=TRUE, transformation=FALSE)

split signature(x="DataTrack"): 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.

Usage:

split(x, f, ...)

Additional Arguments:

  f: the splitting factor.

  ...: all further arguments are ignored.

Examples:

  split(obj, c("a", "a", "b", "c", "a"))

range, ranges signature(x="DataTrack"): return the genomic coordinates for the track as an object of class IRanges.

Usage:

range(x)

ranges(x)

Examples:

  range(obj)

  ranges(obj)

strand signature(x="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.

Usage:

strand(x)

Examples:

  strand(obj)

strand<- signature(x="DataTrack"): replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.

Usage:

strand<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

  strand(obj) <- "+"

feature signature(GdObject="DataTrack"): returns NULL since there is no grouping information for the ranges in a DataTrack.

Usage:

feature(GdObject)

Examples:

  feature(obj)

feature<- signature(gdObject="DataTrack", value="character"): this return the unaltered input object since there is no grouping information for the ranges in a DataTrack.

Usage:

feature<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

feature(obj) <- c("a", "a", "b", "c", "a")

Internal methods:

collapseTrack signature(gdObject="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.

Usage:

collapseTrack(GdObject, diff=.pxResolution(coord="x"))

Additional Arguments:

diff: the minimum pixel width to display, everything below that will be inflated to a width of diff.

Examples:

Gviz:::collapseTrack(obj)

drawGD signature(GdObject="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.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

minBase, maxBase: the coordinate range to plot.

prepare: run method in preparation or in production mode.

subset: subset the object to the visible region or skip the potentially expensive subsetting operation.

...: all further arguments are ignored.

Examples:

Gviz:::drawGD(obj)

Gviz:::drawGD(obj, minBase=1, maxBase=100)

Gviz:::drawGD(obj, prepare=TRUE, subset=FALSE)

drawAxis signature(GdObject="DataTrack"): add a y-axis to the title panel of a track.

Usage:

drawAxis(GdObject, from, to, ...)

Additional Arguments:

from, to: compute axis range from the data within a certain coordinates range only.

...: all further arguments are ignored.

Examples:

Gviz:::drawAxis(obj)

initialize signature(.Object="DataTrack"): initialize the object

show signature(object="DataTrack"): show a human-readable summary of the object

Inherited methods:

drawGrid signature(GdObject="DataTrack"): superpose a grid on top of a track.

Usage:

drawGrid(GdObject, from, to, ...)

Additional Arguments:

from, to: integer scalars, restrict to coordinate range before computing the grid lines.

Examples:

Gviz:::drawGrid(obj)

chromosome signature(GdObject="DataTrack"): return the currently active chromosome for which the track is defined. For consistancy with other Bioconductor packages, the isActiveSeq alias is also provided.

Usage:

chromosome(GdObject)

Examples:

chromosome(obj)

chromosome<- signature(GdObject="DataTrack"): replace the value of the track's active chromosome. This has to be a valid UCSC chromosome identifier or an integer or character scalar that can be reasonably coerced into one, unless options(ucscChromosomeNames=FALSE). For consistancy with other Bioconductor packages, the isActiveSeq<- alias is also provided.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

chromosome(obj) <- "chr12"

start, end, width signature(x="DataTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

start(obj)

end(obj)

width(obj)

start<-, end<-, width<- signature(x="DataTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

start(obj) <- 1:10

end(obj) <- 20:30

width(obj) <- 1

position signature(GdObject="DataTrack"): the arithmetic mean of the track item's coordinates, i.e., (end(obj)-start(obj))/2.

Usage:

position(GdObject)

Examples:

position(obj)

genome signature(x="DataTrack"): return the track's genome.

Usage:

genome(x)

Examples:

genome(obj)

genome<- signature(x="DataTrack"): set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

genome(obj) <- "mm9"

length signature(x="DataTrack"): return the number of items in the track.

Usage:

length(x)

Examples:

length(obj)

coerce signature(from="DataTrack",to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

Examples:

as(obj, "data.frame")

subset signature(x="DataTrack"): subset a NumericTrack by coordinates and sort if necessary.

Usage:

subset(x, from, to, sort=FALSE, drop=TRUE, ...)

Additional Arguments:

from, to: the coordinates range to subset to.

sort: sort the object after subsetting. Usually not necessary.

drop: drop unused regions on the other, non-active chromosomes.w

...: additional arguments are ignored.

Examples:

subset(obj, from=10, to=20, sort=TRUE)

displayPars signature(x="DataTrack",   name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

displayPars(obj, "col")

displayPars signature(x="DataTrack",   name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.

Examples:

displayPars(obj)

getPar signature(x="DataTrack", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

getPar(obj, "col")

getPar signature(x="DataTrack", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.

Examples:

getPar(obj)

displayPars<- signature(x="DataTrack",   value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(x="DataTrack", value="character"): set the single display parameter name to value. Note that display parameters in the DataTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

name: the name of the display parameter to set.

Examples:

setPar(obj, "col", "red")

setPar signature(x="DataTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the DataTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

 setPar(obj, list(col="red", lwd=2))

group signature(GdObject="DataTrack"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

Usage:

group(GdObject)

Examples:

 group(obj)

names signature(x="DataTrack"): return the value of the name slot.

Usage:

names(x)

Examples:

 names(obj)

names<- signature(x="DataTrack", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

 names(obj) <- "foo"

coords signature(ImageMap="DataTrack"): return the coordinates from the internal image map.

Usage:

coords(ImageMap)

Examples:

 coords(obj)

tags signature(x="DataTrack"): return the tags from the internal image map.

Usage:

tags(x)

Examples:

 tags(obj)

Display Parameters

The following display parameters are set for objects of class DataTrack 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.

 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.

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

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 plots, see the 'Details' section in DataTrack for more information.

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.

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

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

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

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

gradient=c("#F7FBFF", "#DEEBF7", "#C6DBEF", "#9ECAE1",   "#6BAED6", "#4292C6", "#2171B5" Character vector. The base colors for the 'gradient' plotting type.

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

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.

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 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 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=FALSE: Logical scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.

notch.frac=0.5: Numeric 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-type plots.

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[[44]]: 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 c("p","l", "b", "a", "s", "g", "r", "S", "smooth", "h 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 mean 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.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of DataTrack objects.

GdObject:

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

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 panels.

cex.axis=NULL: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is computed 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="#0080FF": Integer or character scalar. Default line color 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 color for the y axis, if any.

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

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

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

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

col.title="white": Integer or character scalar. The font 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.

fontface=1: Integer or character scalar. The font face for all text.

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.

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

fontsize=12: Numeric scalar. The font size for all text.

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.

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.grid=1: Numeric scalar. Default line width for grid lines, both when type=="g" in DataTracks and when display parameter grid==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.

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.

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

DataTrack

DisplayPars

GdObject

GRanges

ImageMap

IRanges

NumericTrack

RangeTrack

collapsing

grouping

panel.bwplot

panel.grid

panel.loess

panel.xyplot

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 class and method*

Description

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

Usage

DisplayPars(...)

availableDisplayPars(class)

Arguments

...	All named arguments are stored in the object's environment as individual parameters, regardless of their type.
class	A valid track object class name, or the object itself, in which case the class is derived directly from it.

Details

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 expliciety 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 emvironment needs to be reinstantiated.

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

Value

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

availableDisplayPars returns a list of the default display parameters.

Objects from the Class

Objects can be created using the constructor function DisplayPars.

Slots

pars**:** Object of class "environment", the container for all customization parameters.

Methods

In the following code chunks, obj is considered to be an object of class DisplayPars.

Exported in the name space:

displayPars signature(x="DisplayPars",name="character"): return the value of a subset of
  display parameters, as identified by name.
  Usage:
  displayPars(x, name)
  Examples:
    displayPars(obj, c("foo", "bar"))
    displayPars(obj, "foobar")

displayPars signature(x="DisplayPars",name="missing"): return all available display param-
  eters.
  Usage:
  displayPars(x)
  Examples:
    displayPars(obj)

getPar signature(x="DisplayPars", name="character"): alias for the displayPars method.
  Usage:
  getPar(x, name)
  Examples:
    getPar(obj, "col")

getPar signature(x="DisplayPars", name="missing"): alias for the displayPars method.
  Usage:
  getPar(x)
  Examples:
    getPar(obj)

displayPars<- signature(x="DisplayPars",value="list"): replace or add display parameters as
  provided by the named list items.
  Usage:
  displayPars<-(x, value)
  Examples:
    displayPars(obj) <- list(foo="a", bar=2)

setPar signature(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-assignmnet to the symbol obj is necessary.
  Usage:
  setPar(x, name, value)
  Additional Arguments:
    name: the name of the display parameter to set.
  Examples:
    setPar(obj, "col", "red")

setPar signature(x="DisplayPars", value="list"): set 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-assignmnet to the symbol obj is necessary.
  Examples:
    setPar(obj, list(col="red", lwd=2))

Internal methods:

initialize signature(.Object = "DisplayPars"): initialize the object.

show signature(object = "DisplayPars"): show a human-readable summary of the object.

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 GenomeGraph tracks to a annotation file representation.*

Description

This function is still a bit experimental. 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.

Details

FIXME: Need to support wgl exports as well...

Value

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

Author(s)

Florian Hahne

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 ininitializer has to be called in order to assure that all necessary settings are present.

Objects from the class

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

Slots

dp: Object of class DisplayPars, 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 class ImageMap, 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.

Methods

In the following code chunks, obj is considered to be an object of class GdObject.

Exported in the name space:

displayPars signature(x="GdObject", name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

displayPars(obj, "col")

displayPars signature(x="GdObject", name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.

Examples:

displayPars(obj)

getPar signature(x="GdObject", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

getPar(obj, "col")

getPar signature(x="GdObject", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.

Examples:

getPar(obj)

displayPars<- signature(x="GdObject", value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(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-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

name: the name of the display parameter to set.

Examples:

setPar(obj, "col", "red")

setPar signature(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-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

setPar(obj, list(col="red", lwd=2))

group signature(GdObject="GdObject"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

Usage:

group(GdObject)

Examples:

group(obj)

names signature(x="GdObject"): return the value of the name slot.

Usage:

names(x)

Examples:

names(obj)

names<- signature(x="GdObject", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

names(obj) <- "foo"

coords signature(ImageMap="GdObject"): return the coordinates from the internal image map.

    Usage:

    coords(ImageMap)

    Examples:

     coords(obj)

tags signature(x="GdObject"): return the tags from the internal image map.

    Usage:

    tags(x)

    Examples:

     tags(obj)

subset signature(x="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.

    Usage:

    subset(x, ...)

    Additional Arguments:

     ...: all further arguments are ignored.

    Examples:

     subset(obj)

Internal methods:

drawAxis signature(GdObject="GdObject"): add a y-axis to the title panel of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

    Usage:

    drawAxis(x, ...)

    Additional Arguments:

     ...: all further arguments are ignored.

    Examples:

     Gviz:::drawAxis(obj)

drawGrid signature(GdObject="GdObject"): superpose a grid on top of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

    Usage:

    drawGrid(GdObject, ...)

    Additional Arguments:

     ...: additional arguments are ignored.

    Examples:

     Gviz:::drawGrid(obj)

initialize signature(.Object="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.

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.

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 panels.

cex=1: Numeric scalar. The overall font expansion factor for all text.

cex.axis=NULL: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is computed 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 color 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 color for the y axis, if any.

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

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

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

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

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

collapse=TRUE: Boolean controlling wether 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.

fontcolor="black": Integer or character scalar. The font color for all text.

fontface=1: Integer or character scalar. The font face for all text.

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.

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

fontsize=12: Numeric scalar. The font size for all text.

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.

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 $\text{type}==$"g" in DataTracks and when display parameter $\text{grid}==$TRUE.

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

lwd.grid=1: Numeric scalar. Default line width for grid lines, both when $\text{type}==$"g" in DataTracks and when display parameter $\text{grid}==$TRUE.

min.distance=1: Numeric scalar. The minimum pixel distance before collapsing range items, only if $\text{collapse}==$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.

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.

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.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DisplayPars

GeneRegionTrack

ImageMap

collapsing

DataTrack

panel.grid

plotTracks

settings

GeneRegionTrack-class   *GeneRegionTrack class and methods*

Description

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

Usage

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

Arguments

Since GeneRegionTrack object are essentially just a specific type of AnnotationTrack, their constructors are quite similar. However, in the case of the GeneRegionTrack certain assumptions are made about the type of grouping on different levels (see the Details section for more information). The natural representation for gene model in the Bioconductor world are TranscriptDb objects, and we tried to make it as straight forward as possible to create GeneRegionTracks starting from those. Building the object from individual function arguments is of course still possible.

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:

range   A TranscriptDb object: all the necessary gene model information including exon locations, transcript groupings and associated gene ids are contained in TranscriptDb objects, and the coercion between the two is almost completely automated. If desired, the data to be fetched from the TranscriptDb 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 TranscriptDb object is retained in the GeneRegionTrack.

A GRanges object: the genomic ranges for the GeneRegion track as well as the optional additional elementMetadata 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,\ codesymbol, exon or gene arguments, because it can not be directly encoded in an IRange object. Note that only the former two are mandatory (if not provided explicitly the more or less reasonable default values chromosome=NA and strand=* are used, but not providing information about the gene-to-transcript relationship or the human-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 it will be drawn from the separate chromosome or strand arguments. In addition, the feature, exon, transcript, gene and symbol data can be provided as additional columns. The above comments about potential default values also apply here.

start, end   An integer scalar with the genomic start or end coordinate for the gene model range. If those is missing, the default value will automatically be the smallest (or largest value, respectively) in rstarts for the currently active chromosome. When building a GeneRegionTrack from a TranscriptDb object, these arguments can be used to subset the desired annotation data by genomic coordinates. Please note that in that case the chromosome parameter must also be set.

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.

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. See 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.

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.

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 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. When creating a GeneRegionTrack from a TranscriptDb object, the value of this parameter can be used to subset the data to fetch by 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 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.
...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.

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 transctipt, 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 expected to be coding elements.

Value

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

Objects from the class

Objects can be created using the constructor function GeneRegionTrack.

Slots

start: Object of class "numeric", the start coordinates of the annotation range. The coorrdinates for the individual gene model items are stored in the range slot.

end: Object of class "numeric", the end coordinates of the annotation range. The corrdinates for the individual gene model items are stored in the range slot.

stacking: Object of class "character", inherited from class StackedTrack

stacks: Object of class "environment", inherited from class StackedTrack

range: Object of class GRanges, inherited from class RangeTrack

chromosome: Object of class "character", inherited from class RangeTrack

genome: Object of class "character", inherited from class RangeTrack

dp: Object of class DisplayPars, inherited from class GdObject

name: Object of class "character", inherited from class GdObject

imageMap: Object of class ImageMap, inherited from class GdObject

Extends

Class "AnnotationTrack", directly.

Class "StackedTrack", by class "AnnotationTrack", distance2.

Class "RangeTrack", by class "AnnotationTrack", distance3.

Class "GdObject", by class "AnnotationTrack", distance4.

Methods

In the following code chunks, obj is considered to be an object of class GeneRegionTrack.

Exported in the name space:

group signature(gdObject="GeneRegionTrack"): extract the group membership for all track items.
Usage:
group(GdObject)
Examples:

group(obj)

group<- signature(gdObject="GeneRegionTrack", value="character"): 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.

Usage:

group<-(GdObject, value)

Examples:

group(obj) <- c("a", "a", "b", "c", "a")

identifier signature(gdObject="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.

Usage:

identifier(GdObject, lowest=FALSE)

Additional Arguments:

lowest: return the lowest-level identifier, i.e., the item IDs, or the higher level group IDs which do not have to be unqiue.

Examples:

identifier(obj, lowest=FALSE)

identifier<- signature(gdObject="GeneRegionTrack",value="character"): 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<-.

Usage:

identifier<-(GdObject, value)

Examples:

identifier(obj) <- c("foo", "bar")

exon signature(GdObject="GeneRegionTrack"): Extract the exon identifiers for all exons in the gene models.

Usage:

exon(GdObject)

Examples:

exon(obj)

exon<- signature(GdObject="GeneRegionTrack", value="character"): 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.

Usage:

exon<-(GdObject, value)

Examples:

exon(obj) <- paste("Exon", 1:5)

gene signature(GdObject="GeneRegionTrack"): Extract the gene identifiers for all gene models.

Usage:

gene(GdObject)

Examples:

gene(obj)

gene<- signature(GdObject="GeneRegionTrack",value="character"): replace the gene iden-
tifiers for all gene models. The replacement value must be a character of appropriate length or
another vector that can be coerced into such.

Usage:

gene<-(GdObject, value)

Examples:

  gene(obj) <- paste("Gene", LETTERS[1:5])

symbol signature(GdObject="GeneRegionTrack"): Extract the human-readble gene symbol
for all gene models.

Usage:

symbol(GdObject)

Examples:

  symbol(obj)

symbol<- signature(GdObject="GeneRegionTrack",value="character"): 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.

Usage:

gene<-(GdObject, value)

Examples:

  symbol(obj) <- letters[1:5]

transcript signature(GdObject="GeneRegionTrack"): Extract the transcript identifiers for all
transcripts in the gene models.

Usage:

transcript(GdObject)

Examples:

  transcript(obj)

transcript<- signature(GdObject="GeneRegionTrack",value="character"): replace the tran-
script identifiers for all transcripts in the gene model. The replacement value must be a char-
acter of appropriate length or another vector that can be coerced into such.

Usage:

transcript<-(GdObject, value)

Examples:

  transcript(obj) <- paste("Exon", 1:5)

Internal methods:

coerce signature(from="GeneRegionTrack",to="UCSCData"): coerce to a UCSCData ob-
ject for export to the UCSC genome browser.

Examples:

  as(obj, "UCSCData")

collapseTrack signature(GdObject="GeneRegionTrack"): preprocess the track before plot-
ting. 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.

Usage:

collapseTrack(GdObject, diff=.pxResolution(coord="x"))

Additional Arguments:

diff: the minimum pixel width to display, everything below that will be inflated to a width of diff.

Examples:

Gviz:::collapseTrack(obj)

initialize signature(.Object="GeneRegionTrack"): initialize the object

show signature(object="GeneRegionTrack"): show a human-readable summary of the object

Inherited methods:

drawGD signature(GdObject="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.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

minBase, maxBase: the coordinate range to plot.

prepare: run method in preparation or in production mode.

subset: subset the object to the visible region or skip the potentially expensive subsetting operation.

...: all further arguments are ignored.

Examples:

Gviz:::drawGD(obj)
Gviz:::drawGD(obj, minBase=1, maxBase=100)
Gviz:::drawGD(obj, prepare=TRUE, subset=FALSE)

drawGrid signature(GdObject="GeneRegionTrack"): superpose a grid on top of a track.

Usage:

drawGrid(GdObject, from, to)

Additional Arguments:

from, to: integer scalars, draw grid within a certain coordinates range. This needs to be supplied for the plotting function to know the current genomic coordinates.

Examples:

Gviz:::drawGrid(obj, from=10, to=100)

setStacks signature(GdObject="GeneRegionTrack"): recompute the stacks based on the available space and on the object's track items and stacking settings.

Usage:

setStacks(GdObject, from, to)

Additional Arguments:

from, to: integer scalars, compute stacking within a certain coordinates range. This needs to be supplied for the plotting function to know the current genomic coordinates.

Examples:

Gviz:::setStacks(obj, from=1, to=100)

stacking signature(GdObject="GeneRegionTrack"): return the current stacking type.

    Usage:

    stacking(GdObject)

    Examples:

      stacking(obj)

stacking<- signature(GdObject="GeneRegionTrack",   value="character"): set the object's stacking type to one in c(hide, dense, squish, pack,full).

    Usage:

    stacking<-(GdObject, value)

    Additional Arguments:

      value: replacement value.

    Examples:

      stacking(obj) <- "squish"

stacks signature(GdObject="GeneRegionTrack"): return the stack indices for each track item.

    Usage:

    stacks(GdObject)

    Examples:

      Gviz:::stacks(obj)

[signature(x="GeneRegionTrack", i="ANY", j="ANY",drop="ANY"): subset the items in the GeneRegionTrack 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.

    Additional Arguments:

      i, j: subsetting indices, j is ignored.

      drop: argument is ignored.

    Examples:

      obj[1:5]

chromosome signature(GdObject="GeneRegionTrack"): return the currently active chromosome for which the track is defined. For consistancy with other Bioconductor packages, the isActiveSeq alias is also provided.

    Usage:

    chromosome(GdObject)

    Examples:

      chromosome(obj)

chromosome<- signature(GdObject="GeneRegionTrack"): replace the value of the track's active chromosome. This has to be a valid UCSC chromosome identifier or an integer or character scalar that can be reasonably coerced into one, unless options(ucscChromosomeNames=FALSE). For consistancy with other Bioconductor packages, the isActiveSeq<- alias is also provided.

    Usage:

    chromosome<-(GdObject, value)

    Additional Arguments:

      value: replacement value.

    Examples:

      chromosome(obj) <- "chr12"

start, end, width signature(x="GeneRegionTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

start(obj)

end(obj)

width(obj)

start<-, end<-, width<- signature(x="GeneRegionTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

start(obj) <- 1:10

end(obj) <- 20:30

width(obj) <- 1

position signature(GdObject="GeneRegionTrack"): the arithmetic mean of the track item's coordionates, i.e., (end(obj)-start(obj))/2.

Usage:

position(GdObject)

Examples:

position(obj)

feature signature(GdObject="GeneRegionTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping for details.

Usage:

feature(GdObject)

Examples:

feature(obj)

feature<- signature(gdObject="GeneRegionTrack", value="character"): 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 GeneRegionTrack. See grouping for details.

Usage:

feature<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="GeneRegionTrack"): return the track's genome.

Usage:

genome(x)

Examples:

genome(obj)

genome<- signature(x="GeneRegionTrack"): set the track's genome. Usually this has to be a
valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

genome(obj) <- "mm9"

length signature(x="GeneRegionTrack"): return the number of items in the track.

Usage:

length(x)

Examples:

length(obj)

range signature(x="GeneRegionTrack"): return the genomic coordinates for the track as an
object of class IRanges.

Usage:

range(x)

Examples:

range(obj)

ranges signature(x="GeneRegionTrack"): return the genomic coordinates for the track along
with all additional annotation information as an object of class GRanges.

Usage:

ranges(x)

Examples:

ranges(obj)

split signature(x="GeneRegionTrack"): split a GeneRegionTrack object by an appropriate fac-
tor 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 GeneRegionTrack.

Usage:

split(x, f, ...)

Additional Arguments:

f: the splitting factor.

...: all further arguments are ignored.

Examples:

split(obj, c("a", "a", "b", "c", "a"))

strand signature(x="GeneRegionTrack"): 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.

Usage:

strand(x)

Examples:

strand(obj)

strand<- signature(x="GeneRegionTrack"): replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.

Usage:

strand<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

strand(obj) <- "+"

values signature(x="GeneRegionTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

Usage:

values(x)

Examples:

values(obj)

coerce signature(from="GeneRegionTrack",to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

Examples:

as(obj, "data.frame")

subset signature(x="GeneRegionTrack"): subset a GeneRegionTrack by coordinates and sort if necessary.

Usage:

subset(x, from, to, sort=FALSE, ...)

Additional Arguments:

from, to: the coordinates range to subset to.

sort: sort the object after subsetting. Usually not necessary.

...: additional arguments are ignored.

Examples:

subset(obj, from=10, to=20, sort=TRUE)

displayPars signature(x="GeneRegionTrack", name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

displayPars(obj, "col")

displayPars signature(x="GeneRegionTrack", name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.

Examples:

displayPars(obj)

getPar signature(x="GeneRegionTrack", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

getPar(obj, "col")

getPar signature(x="GeneRegionTrack", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.

Examples:

getPar(obj)

displayPars<- signature(x="GeneRegionTrack",   value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(x="GeneRegionTrack", value="character"): set the single display parameter name to value. Note that display parameters in the GeneRegionTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

name: the name of the display parameter to set.

Examples:

setPar(obj, "col", "red")

setPar signature(x="GeneRegionTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the GeneRegionTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

setPar(obj, list(col="red", lwd=2))

names signature(x="GeneRegionTrack"): return the value of the name slot.

Usage:

names(x)

Examples:

names(obj)

names<- signature(x="GeneRegionTrack", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

names(obj) <- "foo"

coords signature(ImageMap="GeneRegionTrack"): return the coordinates from the internal image map.

Usage:

coords(ImageMap)

Examples:

coords(obj)

tags signature(x="GeneRegionTrack"): return the tags from the internal image map.

Usage:

tags(x)

Examples:

tags(obj)

Display Parameters

The following display parameters are set for objects of class GeneRegionTrack 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.

min.distance=0: Numeric scalar. The minimum pixel distance before collapsing range items, only if \codecollapse==TRUE. See \code\linkcollapsing 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.

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

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

geneSymbols=TRUE: Logical scalar. Use human-readable gene symbols or gene IDs for the transcript annotation.

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

collapseTranscripts=FALSE: Logical scalar. Merge all transcripts of the same gene into one single gene model. Essentially, this will only keep the start location of the first exon and the end location of the last exon from all transcripts of a gene.

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

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of GeneRegionTrack objects.

AnnotationTrack:

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

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

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

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

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

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

fontface=1: Integer scalar. The font face for item identifiers.

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

fontfamily="sans": Character scalar. The font family for item identifiers.

fontsize=12: Numeric scalar. The font size for item identifiers.

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.

rotation=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.

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

StackedTrack:

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.

GdObject:

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

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 panels.

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

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

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

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

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

col.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 font color for the title panels.

collapse=TRUE: Boolean controlling wether 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.

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

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

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

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

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, Steve Lianoglou

See Also

AnnotationTrack

DisplayPars

GdObject

GRanges

ImageMap

IRanges

RangeTrack

StackedTrack

collapsing

DataTrack

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$symbol,
name="Cyp2b10", strand=cyp2b10$strand)

## From a TranscriptDb object
if(require(GenomicFeatures)){
samplefile <- system.file("extdata", "UCSC_knownGene_sample.sqlite", package="GenomicFeatures")
txdb <- loadDb(samplefile)
GeneRegionTrack(txdb)
GeneRegionTrack(txdb, chromosome="chr6", start=300000,
end=350000)
}




## 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=TRUE)
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

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

Arguments

range   Optional GRanges or IRanges object to highlight certain regions on the axis.

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).

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

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 insterad 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 from the class

Objects can be created using the constructor function GenomeAxisTrack.

Slots

range: Object of class GRanges, highlighted on the axis.

dp: Object of class DisplayPars, inherited from class GdObject

name: Object of class "character", inherited from class GdObject

imageMap: Object of class ImageMap, inherited from class GdObject

Extends

Class "GdObject", directly.

Methods

In the following code chunks, obj is considered to be an object of class GenomeAxisTrack.

Exported in the name space:

[signature(x="GenomeAxisTrack"): subset the GRanges object in the range slot. For most applications, the subset method may be more appropriate.

Additional Arguments:

i: subsetting incides.

Examples:

obj[1:5]

start, end, width signature(x="GenomeAxisTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:
start(x)
end(x)
width(x)

Examples:

start(obj)
end(obj)
width(obj)

range signature(x="GenomeAxisTrack"): return the genomic coordinates for the track as an object of class IRanges.

Usage:
range(x)

Examples:

range(obj)

ranges signature(x="GenomeAxisTrack"): return the genomic coordinates for the track along with all additional annotation information as an object of class GRanges.

Usage:
ranges(x)

Examples:

ranges(obj)

strand signature(x="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.

Usage:
strand(x)

Examples:

strand(obj)

values signature(x="GenomeAxisTrack"): return all additional annotation information except for the genomic coordinates for the track items.

Usage:
values(x)

Examples:

values(obj)

subset signature(x="GenomeAxisTrack"): subset a GenomeAxisTrack by coordinates and sort
if necessary.

Usage:

subset(x, from, to, sort=FALSE, ...)

Additional Arguments:

from, to: the coordinates range to subset to.

sort: sort the object after subsetting. Usually not necessary.

...: additional arguments are ignored.

Examples:

subset(obj, from=10, to=20, sort=TRUE)

length signature(x="GenomeAxisTrack"): return the number of items stored in the ranges slot.

Usage:

length(x)

Examples:

length(obj)

Internal methods:

drawGD signature(GdObject="GenomeAxisTrack"): the workhorse function to plot the object.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

minBase, maxBase: the coordinate range to plot.

prepare: run method in preparation or in production mode.

subset: subset the object to the visible region or skip the potentially expensive subsetting
operation.

...: all further arguments are ignored.

Examples:

Gviz:::drawGD(obj)

Gviz:::drawGD(obj, minBase=1, maxBase=100)

Gviz:::drawGD(obj, prepare=TRUE,   subset=FALSE)

collapseTrack signature(GdObject="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.

Usage:

collapseTrack(GdObject, diff=.pxResolution(coord="x"))

Additional Arguments:

diff: the minimum pixel width to display, everything below that will be inflated to a width of
diff.

Examples:

Gviz:::collapseTrack(obj)

initialize signature(.Object="GenomeAxisTrack"): initialize the object

show signature(object="GenomeAxisTrack"): show a human-readable summary of the object

Inherited:

displayPars signature(x="GenomeAxisTrack", name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.
  Usage:
  displayPars(x, name)
  Examples:
    displayPars(obj, "col")

displayPars signature(x="GenomeAxisTrack", name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.
  Examples:
    displayPars(obj)

getPar signature(x="GenomeAxisTrack", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.
  Usage:
  getPar(x, name)
  Examples:
    getPar(obj, "col")

getPar signature(x="GenomeAxisTrack", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.
  Examples:
    getPar(obj)

displayPars<- signature(x="GenomeAxisTrack", value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.
  Usage:
  displayPars<-(x, value)
  Examples:
    displayPars(obj) <- list(col="red", lwd=2)

setPar signature(x="GenomeAxisTrack", value="character"): set the single display parameter name to value. Note that display parameters in the GenomeAxisTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.
  Usage:
  setPar(x, name, value)
  Additional Arguments:
    name: the name of the display parameter to set.
  Examples:
    setPar(obj, "col", "red")

setPar signature(x="GenomeAxisTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the GenomeAxisTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.
  Examples:
    setPar(obj, list(col="red", lwd=2))

group signature(GdObject="GenomeAxisTrack"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

Usage:

group(GdObject)

Examples:

group(obj)

names signature(x="GenomeAxisTrack"): return the value of the name slot.

Usage:

names(x)

Examples:

names(obj)

names<- signature(x="GenomeAxisTrack", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

names(obj) <- "foo"

coords signature(ImageMap="GenomeAxisTrack"): return the coordinates from the internal image map.

Usage:

coords(ImageMap)

Examples:

coords(obj)

tags signature(x="GenomeAxisTrack"): return the tags from the internal image map.

Usage:

tags(x)

Examples:

tags(obj)

drawAxis signature(GdObject="GenomeAxisTrack"): add a y-axis to the title panel of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

Usage:

drawAxis(x, ...)

Additional Arguments:

...: all further arguments are ignored.

Examples:

Gviz:::drawAxis(obj)

drawGrid signature(GdObject="GenomeAxisTrack"): superpose a grid on top of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

Usage:

drawGrid(GdObject, ...)

Additional Arguments:

...: additional arguments are ignored.

Examples:

Gviz:::drawGrid(obj)

Display Parameters

The following display parameters are set for objects of class GenomeAxisTrack 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.

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=0.8: Numeric scalar. The overall font expansion factor for the axis annotation text.

cex.id=0.7: Numeric scalar. The text size for the optional range annotation.

col="darkgray": Character scalar. The color for the axis lines and tickmarks.

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.

showId=FALSE: Logical scalar. Show the optional range highlighting annotation.

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

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.

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.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of GenomeAxisTrack objects.

GdObject:

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

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

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

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

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

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

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

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

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

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

collapse=TRUE: Boolean controlling wether 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=1: Integer or character scalar. The font face for all text.

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.

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.

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

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

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

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

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.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DisplayPars

GdObject

[GRanges](#)

[ImageMap](#)

[IRanges](#)

[RangeTrack](#)

[StackedTrack](#)

[collapsing](#)

[DataTrack](#)

[grouping](#)

[panel.grid](#)

[plotTracks](#)

[settings](#)

Examples

```
## Construct object
axTrack <- GenomeAxisTrack(name="Axis",
range=IRanges::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

IdeogramTrack-class   *IdeogramTrack class and methods*

Description

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

Usage

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

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 centromer 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.

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.

Slots

range**:** Object of class GRanges, inherited from class StackedTrack containing the chromosome band information. This slot is filled automatically by the initializer method.

bandTable**:** Object of class data.frame containing the chromosome band information in the format of UCSC. This slot is filled automatically by the initializer.

bandTable**:** Object of class data.frame containing the chromosome band information for all chromosomes. This slot is filled automatically by the initializer method and only exists to prevent having to redo the rtracklayer query ervery time the chromosome is changed.

chromosome**:** Object of class "character", inherited from class StackedTrack defining the ideogram's chromosome.

genome**:** Object of class "character" inherited from class StackedTrack defining the ideogram's genome.

dp**:** Object of class DisplayPars, inherited from class GdObject

name**:** Object of class "character", inherited from class GdObject

imageMap**:** Object of class ImageMap, inherited from class GdObject

Extends

Class "RangeTrack", directly.

Class "GdObject", by class "RangeTrack", distance 2.

Methods

In the following code chunks, obj is considered to be an object of class IdeogramTrack.

Exported in the name space:

start, end, width, position signature(x/GdObject="IdeogramTrack"): although IdeogramTracks inherit from RangeTrack, the notion of coordinates is not particularly useful. Hence the coordinate methods all return NULL.

Usage:
start(x)
end(x)
width(x)
position(GdObject)

Examples:
start(obj)
end(obj)
width(obj)

start<-, end<-, width<- signature(x="RangeTrack"): although IdeogramTracks inherit from RangeTrack, the notion of coordinates is not particularly useful. Hence the coordinate replacement methods all return the unaltered input object.

Usage:
start<-(x, value)
end<-(x, value)
width<-(x, value)

Additional Arguments:
value: replacement value.

Examples:

    start(obj) <- 1:10
    end(obj) <- 20:30
    width(obj) <- 1

chromosome<- signature(GdObject="IdeogramTrack"): 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. The chromosome band information is updated automatically.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

   value: replacement value.

Examples:

    chromosome(obj) <- "chr12"

genome<- signature(x="IdeogramTrack"): set the track's genome. This has to be a valid UCSC identifier. The chromosome band information is updated automatically.

Usage:

genome<-(x, value)

Additional Arguments:

   value: replacement value.

Examples:

    genome(obj) <- "mm9"

subset signature(x="IdeogramTrack"): subsetting does not make much sense for these object, hence the unalered object is returned.

Usage:

subset(x, ...)

Additional Arguments:

   ...: all further arguments are ignored.

Examples:

    subset(obj)

[signature(x="IdeogramTrack", i="ANY", j="ANY",drop="ANY"): subsetting of IdeogramTrack objects does not make much sense, hence the unaltered input argument is returned.

Additional Arguments:

   i, j: subsetting indices, j is ignored.

   drop: argument is ignored.

Examples:

    obj[1:5]

Internal methods:

drawGD signature(gdObject="IdeogramTrack"): 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.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

minBase, maxBase: the coordinate range to plot.

prepare: run method in preparation or in production mode.

subset: subset the object to the visible region or skip the potentially expensive subsetting operation.

...: all further arguments are ignored.

Examples:

Gviz:::drawGD(obj)

Gviz:::drawGD(obj, minBase=1, maxBase=100)

Gviz:::drawGD(obj, prepare=TRUE,   subset=FALSE)

initialize signature(.Object="IdeogramTrack"): initialize the object.

show signature(object="IdeogramTrack"): show a human-readable summary of the object.

Inherited methods:

chromosome signature(GdObject="IdeogramTrack"): return the chromosome for which the track is defined.

Usage:

chromosome(GdObject)

Examples:

chromosome(obj)

feature signature(GdObject="IdeogramTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping or AnnotationTrack and GeneRegionTrack for details.

Usage:

feature(GdObject)

Examples:

feature(obj)

feature<- signature(gdObject="IdeogramTrack",   value="character"): 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 IdeogramTrack. See grouping or AnnotationTrack and GeneRegionTrack for details.

Usage:

feature<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="IdeogramTrack"): return the track's genome.

Usage:

genome(x)

Examples:

genome(obj)

length signature(x="IdeogramTrack"): return the number of items in the track.

> *Usage:*
>
> length(x)
>
> *Examples:*
>
> length(obj)

range signature(x="IdeogramTrack"): return the genomic coordinates for the track as an object of class IRanges.

> *Usage:*
>
> range(x)
>
> *Examples:*
>
> range(obj)

ranges signature(x="IdeogramTrack"): return the genomic coordinates for the track along with all additional annotation information as an object of class GRanges.

> *Usage:*
>
> ranges(x)
>
> *Examples:*
>
> ranges(obj)

split signature(x="IdeogramTrack"): splitting is not a useful operation for IdeogramTrack objects. *Usage:*

> split(x, f, ...)
>
> *Additional Arguments:*
>
> f: the splitting factor.
>
> ...: all further arguments are ignored.
>
> *Examples:*
>
> split(obj, c("a", "a", "b", "c", "a"))

strand signature(x="IdeogramTrack"): strand information is not relevant for IdeogramTrack objects.

> *Usage:*
>
> strand(x)
>
> *Examples:*
>
> strand(obj)

strand<- signature(x="IdeogramTrack"): strand information is not relevant for IdeogramTrack objects.

> *Usage:*
>
> strand<-(x, value)
>
> *Additional Arguments:*
>
> value: replacement value.
>
> *Examples:*
>
> strand(obj) <- "+"

values signature(x="IdeogramTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

> *Usage:*
>
> values(x)
>
> *Examples:*

values(obj)

coerce signature(from="IdeogramTrack", to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

Examples:

as(obj, "data.frame")

displayPars signature(x="IdeogramTrack",name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

displayPars(obj, "col")

displayPars signature(x="IdeogramTrack", name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.

Examples:

displayPars(obj)

getPar signature(x="IdeogramTrack", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

getPar(obj, "col")

getPar signature(x="IdeogramTrack", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.

Examples:

getPar(obj)

displayPars<- signature(x="IdeogramTrack", value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(x="IdeogramTrack", value="character"): set the single display parameter name to value. Note that display parameters in the IdeogramTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

name: the name of the display parameter to set.

Examples:

setPar(obj, "col", "red")

setPar signature(x="IdeogramTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the IdeogramTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

  setPar(obj, list(col="red", lwd=2))

group signature(GdObject="IdeogramTrack"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

Usage:

group(GdObject)

Examples:

  group(obj)

names signature(x="IdeogramTrack"): return the value of the name slot.

Usage:

names(x)

Examples:

  names(obj)

names<- signature(x="IdeogramTrack", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

  names(obj) <- "foo"

coords signature(ImageMap="IdeogramTrack"): return the coordinates from the internal image map.

Usage:

coords(ImageMap)

Examples:

  coords(obj)

tags signature(x="IdeogramTrack"): return the tags from the internal image map.

Usage:

tags(x)

Examples:

  tags(obj)

drawAxis signature(GdObject="IdeogramTrack"): add a y-axis to the title panel of a track if necessary. For IdeogramTrack objects this does not plot anything and returns NULL.

Usage:

drawAxis(x, ...)

Additional Arguments:

  ...: all further arguments are ignored.

Examples:

  Gviz:::drawAxis(obj)

drawGrid signature(GdObject="IdeogramTrack"): superpose a grid on top of a track if necessary. For IdeogramTrack objects this does not plot anything and returns NULL.

Usage:

drawGrid(GdObject, ...)

Additional Arguments:

...: additional arguments are ignored.

Examples:

Gviz:::drawGrid(obj)

Display Parameters

The following display parameters are set for objects of class IdeogramTrack 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.

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.

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.

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

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

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

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 be overridden in the plotTracks function.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of IdeogramTrack objects.

GdObject:

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

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

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

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

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

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

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

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

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

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

collapse=TRUE: Boolean controlling wether to collapse the content of the track to accomodate the minimum current device resolution. See collapsing for details.

fontface=1: Integer or character scalar. The font face for all text.

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.

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.

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

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

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

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

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.

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

AnnotationTrack

DisplayPars

GdObject

GeneRegionTrack

GRanges

ImageMap

IRanges

RangeTrack

StackedTrack

clearSessionCache

collapsing

DataTrack

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 class and methods*

Description

HTML image map information for annotation tracks.

Objects from the Class

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

Slots

coords**:** Object of class "matrix", the image map coordinats.

tags**:** Object of class "list", the individual HTML tags for the image map.

Extends

Class "ImageMapOrNULL", directly.

Methods

coords signature(ImageMap="ImageMap"): return the coordinates from the image map.
   Usage:
   coords(ImageMap)
   Examples:
   coords(obj)

tags signature(x="ImageMap"): return the tags from the image map.
   Usage:
   tags(x)
   Examples:
   tags(obj)

Author(s)

Florian Hahne

| 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.

Objects from the class

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

Slots

range: Object of class GRanges, inherited from class RangeTrack

chromosome: Object of class "character", inherited from class RangeTrack

genome: Object of class "character", inherited from class RangeTrack

dp: Object of class DisplayPars, inherited from class GdObject

name: Object of class "character", inherited from class GdObject

imageMap: Object of class ImageMap, inherited from class GdObject

Extends

Class "RangeTrack", directly.

Class "GdObject", by class "RangeTrack", distance 2.

Methods

Internal methods:

drawAxis signature(GdObject="NumericTrack"): add a y-axis to the title panel of a track.
   Usage:
   drawAxis(x, from, to, ...)
   Additional Arguments:

   from, to: integer scalars, restrict to coordinate range before computing the axis ranges.
   ...: additional arguments are ignored.

   Examples:

   Gviz:::drawAxis(obj)

drawGrid signature(GdObject="NumericTrack"): superpose a grid on top of a track.
   Usage:
   drawGrid(GdObject, from, to, ...)
   Additional Arguments:

   from, to: integer scalars, restrict to coordinate range before computing the grid lines.

   Examples:

   Gviz:::drawGrid(obj)

initialize signature(.Object="NumericTrack"): initialize the object.

Inherited methods:

[signature(x="NumericTrack", i="ANY", j="ANY",drop="ANY"): subset the items in the NumericTrack 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.

Additional Arguments:

  i, j: subsetting indices, j is ignored.

  drop: argument is ignored.

Examples:

  obj[1:5]

chromosome signature(GdObject="NumericTrack"): return the chromosome for which the track is defined.

Usage:

chromosome(GdObject)

Examples:

  chromosome(obj)

chromosome<- signature(GdObject="NumericTrack"): 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.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

  value: replacement value.

Examples:

  chromosome(obj) <- "chr12"

start, end, width signature(x="NumericTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

  start(obj)

  end(obj)

  width(obj)

start<-, end<-, width<- signature(x="NumericTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

      start(obj) <- 1:10

      end(obj) <- 20:30

      width(obj) <- 1

position signature(GdObject="NumericTrack"): the arithmetic mean of the track item's coordionates, i.e., (end(obj)-start(obj))/2.

    Usage:

    position(GdObject)

    Examples:

     position(obj)

feature signature(GdObject="NumericTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping or AnnotationTrack and GeneRegionTrack for details.

    Usage:

    feature(GdObject)

    Examples:

     feature(obj)

feature<- signature(gdObject="NumericTrack",   value="character"): 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 NumericTrack. See grouping or AnnotationTrack and GeneRegionTrack for details.

    Usage:

    feature<-(GdObject, value)

    Additional Arguments:

     value: replacement value.

    Examples:

     feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="NumericTrack"): return the track's genome.

    Usage:

    genome(x)

    Examples:

     genome(obj)

genome<- signature(x="NumericTrack"): set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.

    Usage:

    genome<-(x, value)

    Additional Arguments:

     value: replacement value.

    Examples:

     genome(obj) <- "mm9"

length signature(x="NumericTrack"): return the number of items in the track.

    Usage:

    length(x)

    Examples:

length(obj)

range signature(x="NumericTrack"): return the genomic coordinates for the track as an object of class IRanges.

Usage:

range(x)

Examples:

range(obj)

ranges signature(x="NumericTrack"): return the genomic coordinates for the track along with all additional annotation information as an object of class GRanges.

Usage:

ranges(x)

Examples:

ranges(obj)

split signature(x="NumericTrack"): split a NumericTrack 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 NumericTrack.

Usage:

split(x, f, ...)

Additional Arguments:

f: the splitting factor.

...: all further arguments are ignored.

Examples:

split(obj, c("a", "a", "b", "c", "a"))

strand signature(x="NumericTrack"): 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.

Usage:

strand(x)

Examples:

strand(obj)

strand<- signature(x="NumericTrack"): replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.

Usage:

strand<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

strand(obj) <- "+"

values signature(x="NumericTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

Usage:

values(x)

Examples:

values(obj)

coerce signature(from="NumericTrack",to="data.frame"): coerce the GRanges object in the
   range slot into a regular data.frame.

   Examples:

   as(obj, "data.frame")

subset signature(x="NumericTrack"): subset a NumericTrack by coordinates and sort if nec-
   essary.

   Usage:

   subset(x, from, to, sort=FALSE, ...)

   Additional Arguments:

   from, to: the coordinates range to subset to.

   sort: sort the object after subsetting. Usually not necessary.

   ...: additional arguments are ignored.

   Examples:

   subset(obj, from=10, to=20, sort=TRUE)

displayPars signature(x="NumericTrack",   name="character"): list the value of the dis-
   play parameter name. See settings for details on display parameters and customization.

   Usage:

   displayPars(x, name)

   Examples:

   displayPars(obj, "col")

displayPars signature(x="NumericTrack",   name="missing"): list the value of all available
   display parameters. See settings for details on display parameters and customization.

   Examples:

   displayPars(obj)

getPar signature(x="NumericTrack", name="character"): alias for the displayPars method.
   See settings for details on display parameters and customization.

   Usage:

   getPar(x, name)

   Examples:

   getPar(obj, "col")

getPar signature(x="NumericTrack", name="missing"): alias for the displayPars method.
   See settings for details on display parameters and customization.

   Examples:

   getPar(obj)

displayPars<- signature(x="NumericTrack",   value="list"): set display parameters using
   the values of the named list in value. See settings for details on display parameters and
   customization.

   Usage:

   displayPars<-(x, value)

   Examples:

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

setPar signature(x="NumericTrack", value="character"): set the single display parameter name to value. Note that display parameters in the NumericTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

  name: the name of the display parameter to set.

Examples:

  setPar(obj, "col", "red")

setPar signature(x="NumericTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the NumericTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

  setPar(obj, list(col="red", lwd=2))

group signature(GdObject="NumericTrack"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

Usage:

group(GdObject)

Examples:

  group(obj)

names signature(x="NumericTrack"): return the value of the name slot.

Usage:

names(x)

Examples:

  names(obj)

names<- signature(x="NumericTrack", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

  names(obj) <- "foo"

coords signature(ImageMap="NumericTrack"): return the coordinates from the internal image map.

Usage:

coords(ImageMap)

Examples:

  coords(obj)

tags signature(x="NumericTrack"): return the tags from the internal image map.

Usage:

tags(x)

Examples:

  tags(obj)

Display Parameters

No formal display parameters are defined for objects of class NumericTrack.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of NumericTrack objects.

GdObject:

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

   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 panels.

   cex=1: Numeric scalar. The overall font expansion factor for all text.

   cex.axis=NULL: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is computed 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="#0080FF": Integer or character scalar. Default line color 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 color for the y axis, if any.

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

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

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

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

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

   collapse=TRUE: Boolean controlling wether 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.

   fontcolor="black": Integer or character scalar. The font color for all text.

   fontface=1: Integer or character scalar. The font face for all text.

   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.

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

   fontsize=12: Numeric scalar. The font size for all text.

   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.

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

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

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

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

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.

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.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DisplayPars

GdObject

GeneRegionTrack

GRanges

ImageMap

IRanges

RangeTrack

collapsing

DataTrack

grouping

panel.grid

plotTracks

settings

| plotTracks | *The main plotting function for one or several GenomeGraph 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, execpt 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)
```

Arguments

trackList   A list of GenomeGraph 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   Charactar 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 lenght 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 accomodate 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.

Details

GenomeGraph 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 GenomeGraph 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

RangeTrack

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.

Objects from the class

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

Slots

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 usualy 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 subclasses this has to be valid UCSC genome identifier, however this may not always be formally checked upon object instantiation.

dp**:** Object of class DisplayPars, inherited from class GdObject.

name**:** Object of class "character", inherited from class GdObject

imageMap**:** Object of class ImageMap, inherited from class GdObject

Extends

Class "GdObject", directly.

Methods

In the following code chunks, obj is considered to be an object of class RangeTrack.

Exported in the name space:

[signature(x="RangeTrack", i="ANY", j="ANY",drop="ANY"): subset the items in the RangeTrack object. This is essentially similar to subsetting of the GRanges object in the range slot. For most applications, the subset method may be more appropriate.

Additional Arguments:

i, j: subsetting indices, j is ignored.

drop: argument is ignored.

Examples:

obj[1:5]

chromosome signature(GdObject="RangeTrack"): return the chromosome for which the track is defined.

Usage:
chromosome(GdObject)

Examples:

chromosome(obj)

chromosome<- signature(GdObject="RangeTrack"): 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.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

  value: replacement value.

Examples:

  chromosome(obj) <- "chr12"

start, end, width signature(x="RangeTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

  start(obj)
  end(obj)
  width(obj)

start<-, end<-, width<- signature(x="RangeTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

  start(obj) <- 1:10
  end(obj) <- 20:30
  width(obj) <- 1

position signature(GdObject="RangeTrack"): the arithmetic mean of the track item's coordionates, i.e., (end(obj)-start(obj))/2.

Usage:

position(GdObject)

Examples:

  position(obj)

feature signature(GdObject="RangeTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping or AnnotationTrack and GeneRegionTrack for details.

Usage:

feature(GdObject)

Examples:

  feature(obj)

feature<- signature(gdObject="RangeTrack",   value="character"): 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.

Usage:

feature<-(GdObject, value)

Additional Arguments:

  value: replacement value.

Examples:

  feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="RangeTrack"): return the track's genome.

Usage:

genome(x)

Examples:

  genome(obj)

genome<- signature(x="RangeTrack"): set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

  genome(obj) <- "mm9"

length signature(x="RangeTrack"): return the number of items in the track.

Usage:

length(x)

Examples:

  length(obj)

range signature(x="RangeTrack"): return the genomic coordinates for the track as an object of class IRanges.

Usage:

range(x)

Examples:

  range(obj)

ranges signature(x="RangeTrack"): return the genomic coordinates for the track along with all additional annotation information as an object of class GRanges.

Usage:

ranges(x)

Examples:

  ranges(obj)

split signature(x="RangeTrack"): 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.

Usage:

split(x, f, ...)

Additional Arguments:

      f: the splitting factor.

      ...: all further arguments are ignored.

    Examples:

     split(obj, c("a", "a", "b", "c", "a"))

strand signature(x="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.

    Usage:

    strand(x)

    Examples:

     strand(obj)

strand<- signature(x="RangeTrack"): replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.

    Usage:

    strand<-(x, value)

    Additional Arguments:

     value: replacement value.

    Examples:

     strand(obj) <- "+"

values signature(x="RangeTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

    Usage:

    values(x)

    Examples:

     values(obj)

min signature(...="RangeTrack"): return the start position for the leftmost range item.

    Examples:

     min(obj)

max signature(...="RangeTrack"): return the end position for the rightmost range item.

    Examples:

     max(obj)

coerce signature(from="RangeTrack", to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

    Examples:

     as(obj, "data.frame")

subset signature(x="RangeTrack"): subset a RangeTrack by coordinates and sort if necessary.

    Usage:

    subset(x, from, to, sort=FALSE, ...)

    Additional Arguments:

     from, to: the coordinates range to subset to.

     sort: sort the object after subsetting. Usually not necessary.

     ...: additional arguments are ignored.

    Examples:

subset(obj, from=10, to=20, sort=TRUE)

Internal methods:

initialize signature(.Object="RangeTrack"): initialize the object.

Inherited methods:

displayPars signature(x="RangeTrack", name="character"): list the value of the display
parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

displayPars(obj, "col")

displayPars signature(x="RangeTrack", name="missing"): list the value of all available dis-
play parameters. See settings for details on display parameters and customization.

Examples:

displayPars(obj)

getPar signature(x="RangeTrack", name="character"): alias for the displayPars method.
See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

getPar(obj, "col")

getPar signature(x="RangeTrack", name="missing"): alias for the displayPars method. See
settings for details on display parameters and customization.

Examples:

getPar(obj)

displayPars<- signature(x="RangeTrack", value="list"): set display parameters using the val-
ues of the named list in value. See settings for details on display parameters and customiza-
tion.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(x="RangeTrack", value="character"): set the single display parameter name
to value. Note that display parameters in the RangeTrack class are pass-by-reference, so no
re-assignmnet to the symbol obj is necessary. See settings for details on display parameters
and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

name: the name of the display parameter to set.

Examples:

setPar(obj, "col", "red")

setPar signature(x="RangeTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the RangeTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

> *Examples:*

> setPar(obj, list(col="red", lwd=2))

group signature(GdObject="RangeTrack"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

> *Usage:*

> group(GdObject)

> *Examples:*

> group(obj)

names signature(x="RangeTrack"): return the value of the name slot.

> *Usage:*

> names(x)

> *Examples:*

> names(obj)

names<- signature(x="RangeTrack", value="character"): set the value of the name slot.

> *Usage:*

> names<-(x, value)

> *Examples:*

> names(obj) <- "foo"

coords signature(ImageMap="RangeTrack"): return the coordinates from the internal image map.

> *Usage:*

> coords(ImageMap)

> *Examples:*

> coords(obj)

tags signature(x="RangeTrack"): return the tags from the internal image map.

> *Usage:*

> tags(x)

> *Examples:*

> tags(obj)

drawAxis signature(GdObject="RangeTrack"): add a y-axis to the title panel of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

> *Usage:*

> drawAxis(x, ...)

> *Additional Arguments:*

> ...: all further arguments are ignored.

> *Examples:*

> Gviz:::drawAxis(obj)

drawGrid signature(GdObject="RangeTrack"): superpose a grid on top of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

Usage:

drawGrid(GdObject, ...)

Additional Arguments:

...: additional arguments are ignored.

Examples:

Gviz:::drawGrid(obj)

Display Parameters

No formal display parameters are defined for objects of class RangeTrack.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of RangeTrack objects.

GdObject:

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

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 panels.

cex=1: Numeric scalar. The overall font expansion factor for all text.

cex.axis=NULL: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is computed 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="#0080FF": Integer or character scalar. Default line color 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 color for the y axis, if any.

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

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

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

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

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

collapse=TRUE: Boolean controlling wether 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.

fontcolor="black": Integer or character scalar. The font color for all text.

fontface=1: Integer or character scalar. The font face for all text.

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.

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

fontsize=12: Numeric scalar. The font size for all text.

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.

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

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

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

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

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.

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.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DataTrack

DisplayPars

GdObject

GeneRegionTrack

GRanges

ImageMap

IRanges

collapsing

grouping

panel.grid

plotTracks

settings

SequenceTrack-class   *SequenceTrack class and methods*

Description

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

Usage

SequenceTrack(sequence = NULL, chromosome = NULL, genome, name = "Sequence", ...)

Arguments

sequence	Either an object of class DNAStringSet or an object of class BSgenome. In the former case, the individual DNAStrings are considered to be the different chromosome sequences. In the latter case, the Gviz package tries to follow the BSgenome philosophy in that the respective chromosome sequences are only realized once they are first accessed.
chromosome	The currently active chromosome of the track. 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 sequences for the respective chromosomes are indeed part of the object. If not provided here, the constructor will set it to the first available sequence. 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. 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.
name	Character scalar of the track's name used in the title panel when plotting.
...	Additional items which will all be interpreted as further display parameters. See settings and the "Display Parameters" section below for details.

Value

The return value of the constructor function is a new object of class SequenceDNAStringSetTrack or of class SequenceBSgenomeTrack 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.

Objects from the class

Objects can be created using the constructor function SequenceTrack.

details

Depending on the available space the class will use different options to plot a sequence. If single letters can be accomodated without overplotting those will be show. Otherwise, colored boxes will be used to indicate letters, and if there is not enough horizontal room to show those, a simple line will indicate presence of a sequence. The min.width and fontsize display parameters directly control this behaviour. Each of the five possible nucleotides (G, A, T, C, and N) will be endoded in a separate color. As default we use the colors suggested in the biovizBase package, but a user is free to set their own color scheme by providing a named character vector with color as display parameter fontcolor, with names equal to the five possible bases.

Slots

chromosome**:** Object of class "character", the chromosome on which the track is defined. There can only be a single chromosome for one track. 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. This should be a valid UCSC genome identifier, however this may not always be formally checked upon object instantiation.

dp**:** Object of class DisplayPars, inherited from class GdObject.

name**:** Object of class "character", inherited from class GdObject

imageMap**:** Object of class ImageMap, inherited from class GdObject

Extends

Class "GdObject", directly.

Methods

In the following code chunks, obj is considered to be an object inheriting from class SequenceTrack.

Exported in the name space:

chromosome signature(GdObject="SequenceTrack"): return the chromosome for which the track is defined.
   Usage:
   chromosome(GdObject)
   Examples:
     chromosome(obj)

chromosome<- signature(GdObject="SequenceTrack"): 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.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

chromosome(obj) <- "chr12"

genome signature(x="SequenceTrack"): return the track's genome.

Usage:

genome(x)

Examples:

genome(obj)

genome<- signature(x="SequenceTrack"): set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

genome(obj) <- "mm9"

length signature(x="SequenceTrack"): return the number of nucleotides in the track's sequence.

Usage:

length(x)

Examples:

length(obj)

seqnames signature(x="SequenceTrack"): return the names (i.e., the chromosome) of the sequences contained in the object.

Usage:

values(x)

Examples:

seqnames(obj)

subseq signature(x="SequenceTrack"): Extract a sub-sequence from the track.

Usage:

subseq(x, start=NA, end=NA, width=NA)

Additional Arguments:

start: the start coordinate for the sub-sequence.

end: the end coordinate for the sub-sequence.

width: the width of the sub-sequence.

Examples:

subseq(obj, 1, 10)

Internal methods:

initialize signature(.Object="SequenceTrack"): initialize the object.

Inherited methods:

displayPars signature(x="SequenceTrack",   name="character"): list the value of the
    display parameter name. See settings for details on display parameters and customization.
    Usage:
    displayPars(x, name)
    Examples:

      displayPars(obj, "col")

displayPars signature(x="SequenceTrack", name="missing"): list the value of all available
    display parameters. See settings for details on display parameters and customization.
    Examples:

      displayPars(obj)

getPar signature(x="SequenceTrack", name="character"): alias for the displayPars method.
    See settings for details on display parameters and customization.
    Usage:
    getPar(x, name)
    Examples:

      getPar(obj, "col")

getPar signature(x="SequenceTrack", name="missing"): alias for the displayPars method.
    See settings for details on display parameters and customization.
    Examples:

      getPar(obj)

displayPars<- signature(x="SequenceTrack", value="list"): set display parameters using the
    values of the named list in value. See settings for details on display parameters and cus-
    tomization.
    Usage:
    displayPars<-(x, value)
    Examples:

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

setPar signature(x="SequenceTrack", value="character"): set the single display parameter
    name to value. Note that display parameters in the SequenceTrack class are pass-by-reference,
    so no re-assignmnet to the symbol obj is necessary. See settings for details on display param-
    eters and customization.
    Usage:
    setPar(x, name, value)
    Additional Arguments:

      name: the name of the display parameter to set.

    Examples:

      setPar(obj, "col", "red")

setPar signature(x="SequenceTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the SequenceTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

> *Examples:*
>
>   setPar(obj, list(col="red", lwd=2))

names signature(x="SequenceTrack"): return the value of the name slot.

> *Usage:*
>
> names(x)
>
> *Examples:*
>
>   names(obj)

names<- signature(x="SequenceTrack", value="character"): set the value of the name slot.

> *Usage:*
>
> names<-(x, value)
>
> *Examples:*
>
>   names(obj) <- "foo"

coords signature(ImageMap="SequenceTrack"): return the coordinates from the internal image map.

> *Usage:*
>
> coords(ImageMap)
>
> *Examples:*
>
>   coords(obj)

tags signature(x="SequenceTrack"): return the tags from the internal image map.

> *Usage:*
>
> tags(x)
>
> *Examples:*
>
>   tags(obj)

drawAxis signature(GdObject="SequenceTrack"): add a y-axis to the title panel of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

> *Usage:*
>
> drawAxis(x, ...)
>
> *Additional Arguments:*
>
>   ...: all further arguments are ignored.
>
> *Examples:*
>
>   Gviz:::drawAxis(obj)

drawGrid signature(GdObject="SequenceTrack"): superpose a grid on top of a track if necessary. Unless overwritten in one of the sub-classes this usualy does not plot anything and returns NULL.

> *Usage:*
>
> drawGrid(GdObject, ...)
>
> *Additional Arguments:*
>
>   ...: additional arguments are ignored.
>
> *Examples:*
>
>   Gviz:::drawGrid(obj)

Display Parameters

The following display parameters are set for objects of class SequenceTrack upon instantiation

size=null: Numeric scalar. The size of the track item. Defaults to auto-detect the size based on the other parameter settings.

fontcolor=getBioColor("DNA_BASES_N"): Character vector. The colors used for the 5 possible nucleotides (G, A, T, C, N). Defaults to use colors as defined in the biovizBase package.

fontsize=10: Numeric scalar. Controls the size of the sequence and thus also the level of plotable details.

fontface=2: Numeric scalar. The face of the font.

lwd=2: Numeric scalar. The width of the line when no indiviual letters can be plotted due to size limitations.

col="darkgray": Character scalar. The color of the line when no indiviual letters can be plotted due to size limitations.

min.width=2: Numeric scalar. The minimum width of the colored boxes that are drawn when no indiviual letters can be plotted due to size limitations.

showTitle=FALSE: Logical scalar. Do not show a title panel by default.

background.title="transparent": Character scalar. Make the title panel transparent by default.

noLetters=FALSE: Logical scalar. Always plot colored boxes (or a line) regardles of the available space.

add53=FALSE: Logical scalar. Add a direction indicator.

add53=FALSE: Logical scalar. Plot the sequence complement.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of SequenceTrack objects.

GdObject:

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

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

cex=1: Numeric scalar. The overall font expansion factor for all text.

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

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

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

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

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

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

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

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

collapse=TRUE: Boolean controlling wether to collapse the content of the track to acco-
modate the minimum current device resolution. See collapsing for details.

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

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.

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.

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

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

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

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.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DataTrack

DisplayPars

GdObject

GeneRegionTrack

GRanges

ImageMap

IRanges

BSgenome

DNAStringSet

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.

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.

Display Parameters

GenomeAxisTrack :

cex.id=0.7: Numeric scalar. The text size for the optional range annotation.

fontsize=10: Numeric scalar. Font size for the axis annotation text in points.

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

fontcolor="#808080": Character scalar. The font color for the axis annotation text.

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.

lwd=2: Numeric scalar. The line width for the axis elementes.

add35=FALSE: Logical scalar. Add 3' to 5' direction indicators.

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

add53=FALSE: Logical scalar. Add 5' to 3' direction indicators.

littleTicks=FALSE: Logical scalar. Add more fine-grained tick marks.

distFromAxis=1: Numeric scalar. Control the distance of the axis annotation from the tick marks.

col.id="white": Character scalar. The text color for the optional range annotation.

labelPos="alternating": Character vector, one in "alternating", "revAlternating", "above" or "below". The vertical positioning of the axis labels.

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.

cex=0.8: Numeric scalar. The overall font expansion factor for the axis annotation text.

fill.range="cornsilk3": Character scalar. The fill color for highlighted regions on the axis.

showId=FALSE: Logical scalar. Show the optional range highlighting annotation.

col.range="cornsilk4": Character scalar. The border color for highlighted regions on the axis.

col="darkgray": Character scalar. The color for the axis lines and tickmarks.

Inherited from class GdObject:

fontface=1: Integer or character scalar. The font face for all text.

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see panel.grid for details.

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

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

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.

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

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

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

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

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

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

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.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.

collapse=TRUE: Boolean controlling wether to collapse the content of the track to accomodate the minimum current device resolution. See collapsing for details.

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

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

lineheight=1: Numeric scalar. The font line height for all text.

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

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

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

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.

col.title="white": Integer or character scalar. The font color 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.

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

fontfamily="sans": Integer or character scalar. The font family for all text.

DataTrack :

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

notch=FALSE: Logical scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.

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

box.width=NULL: Numeric scalar. Parameter controlling the boxplot appearance. See panel.bwplot 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.

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

gradient=c("#F7FBFF", "#DEEBF7", "#C6DBEF", "#9ECAE1", "#6BAED6", "#4292C6", "#": Character vector. The base colors for the 'gradient' plotting type.

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

degree=1: Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See panel.loess for details.

type="p": Character vector. The plot type, one or several in c("p","l", "b", "a", "s", "g", "r", "S", "smoo See 'Details' section in DataTrack for more information on the individual plotting types.

notch.frac=0.5: Numeric scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.

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

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

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

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

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.

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

span=0.2: Numeric scalar. Parameter controlling the loess calculation for smooth and mountain-type plots. See panel.loess 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.

box.ratio=1: Numeric scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.

varwidth=FALSE: Logical scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.

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

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

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

stats=X[[44]]: Function. Parameter controlling the boxplot appearance. See panel.bwplot for details.

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

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

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

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

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

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.

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

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

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

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

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

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

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

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

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.

coef=1.5: Numeric scalar. Parameter controlling the boxplot appearance. See panel.bwplot for details.

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

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.

Inherited from class GdObject:

fontface=1: Integer or character scalar. The font face for all text.

fontsize=12: Numeric scalar. The font size for all text.

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see panel.grid for details.

fontcolor="black": Integer or character scalar. The font color for all text.

lwd=1: Numeric scalar. Default line width 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 color for the y axis, if any.

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

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

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

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

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

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

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.

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

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

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

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

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.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.

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

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

lineheight=1: Numeric scalar. The font line height for all text.

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

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

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

col="#0080FF": Integer or character scalar. Default line color 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.

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

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

fontfamily="sans": Integer or character scalar. The font family for all text.

IdeogramTrack :

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

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

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

size=NULL: Numeric scalar. The relative size of the track. Defaults to automatic size setting. Can be overridden in the plotTracks function.

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

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

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

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

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

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

Inherited from class GdObject:

fontface=1: Integer or character scalar. The font face for all text.

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see panel.grid for details.

lwd=1: Numeric scalar. Default line width 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 color for the y axis, if any.

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

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.

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

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

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

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

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

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

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.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.

collapse=TRUE: Boolean controlling wether to collapse the content of the track to accomodate the minimum current device resolution. See collapsing for details.

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

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

lineheight=1: Numeric scalar. The font line height for all text.

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

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

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.

col.title="white": Integer or character scalar. The font color 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.

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

fontfamily="sans": Integer or character scalar. The font family for all text.

AnnotationTrack :

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

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

fontsize=12: Numeric scalar. The font size for item identifiers.

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

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

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

fontfamily="sans": Character scalar. The font family for item identifiers.

size=1: Numeric scalar. The relative size of the track. Can be overridden in the plotTracks function.

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

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.

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

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

fontface=1: Integer scalar. The font face for item identifiers.

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

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

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

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

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

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

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

Inherited from class GdObject:

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see panel.grid for details.

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

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

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

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

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

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

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

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.

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

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

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

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.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.

collapse=TRUE: Boolean controlling wether to collapse the content of the track to accomodate the minimum current device resolution. See collapsing for details.

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

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

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

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.

col.title="white": Integer or character scalar. The font color 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.

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

GeneRegionTrack :

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

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

geneSymbols=TRUE: Logical scalar. Use human-readable gene symbols or gene IDs for the transcript annotation.

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

Inherited from class AnnotationTrack:

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

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

fontsize=12: Numeric scalar. The font size for item identifiers.

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

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

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

fontfamily="sans": Character scalar. The font family for item identifiers.

size=1: Numeric scalar. The relative size of the track. Can be overridden in the plotTracks function.

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

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.

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

fontface=1: Integer scalar. The font face for item identifiers.

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

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

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

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

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

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

Inherited from class GdObject:

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see panel.grid for details.

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

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

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

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

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

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

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

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.

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

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

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

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.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.

collapse=TRUE: Boolean controlling wether to collapse the content of the track to accomodate the minimum current device resolution. See collapsing for details.

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

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

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

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.

col.title="white": Integer or character scalar. The font color 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.

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

BiomartGeneRegionTrack :

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

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

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

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

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

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

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

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

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

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

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'.

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

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

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

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

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

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

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

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

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

Inherited from class GeneRegionTrack:

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

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

geneSymbols=TRUE: Logical scalar. Use human-readable gene symbols or gene IDs for the transcript annotation.

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

Inherited from class AnnotationTrack:

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

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

fontsize=12: Numeric scalar. The font size for item identifiers.

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

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

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

fontfamily="sans": Character scalar. The font family for item identifiers.

size=1: Numeric scalar. The relative size of the track. Can be overridden in the plotTracks function.

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

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.

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

fontface=1: Integer scalar. The font face for item identifiers.

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

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

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

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

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

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

Inherited from class GdObject:

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see panel.grid for details.

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

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

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

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

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

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

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

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.

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

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

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

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.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.

collapse=TRUE: Boolean controlling wether to collapse the content of the track to accomodate the minimum current device resolution. See collapsing for details.

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

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

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

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.

col.title="white": Integer or character scalar. The font color 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.

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

AlignedReadTrack :

detail="coverage": the amount of detail to plot the data. Either coverage to show the coverage only, or reads to show individual reads. For large data sets the latter can be very inefficient. Please note that reads is only available when the object has been created with option coverageOnly=FALSE.

type="histogram": the plot type, one or several in c("p","l", "b", "a", "s", "g", "r", "S", "smooth",   "his See the 'Details' section in DataTrack for more information on the individual plotting types.

size=NULL: the relative size of the track. Defaults to size selection based on the underlying data. Can be overridden in the plotTracks function.

collapse=FALSE: collapse overlapping ranges and aggregate the underlying data.

fill="#0080ff": the fill color for the coverage indicator.

Inherited from class GdObject:

fontface=1: Integer or character scalar. The font face for all text.

fontsize=12: Numeric scalar. The font size for all text.

h=-1: Integer scalar. Parameter controlling the number of horizontal grid lines, see panel.grid for details.

fontcolor="black": Integer or character scalar. The font color for all text.

lwd=1: Numeric scalar. Default line width 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 color for the y axis, if any.

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

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

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

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

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

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

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.

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

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

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

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

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.

cex=1: Numeric scalar. The overall font expansion factor for all text.

v=-1: Integer scalar. Parameter controlling the number of vertical grid lines, see panel.grid for details.

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

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

lineheight=1: Numeric scalar. The font line height for all text.

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

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

col="#0080FF": Integer or character scalar. Default line color 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.

col.title="white": Integer or character scalar. The font color 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.

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

fontfamily="sans": Integer or character scalar. The font family for all text.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DataTrack

DisplayPars

GdObject

availableDisplayPars

collapsing

displayPars

grouping

panel.bwplot

panel.grid

panel.loess

panel.xyplot

plotTracks

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.

Objects from the Class

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

Slots

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.

range**:** Object of class GRanges, inherited from class RangeTrack

chromosome**:** Object of class "character", inherited from class RangeTrack

genome**:** Object of class "character", inherited from class RangeTrack

dp**:** Object of class DisplayPars, inherited from class GdObject

name**:** Object of class "character", inherited from class GdObject

imageMap**:** Object of class ImageMap, inherited from class GdObject

Extends

Class "RangeTrack", directly.

Class "GdObject", by class "RangeTrack", distance 2.

Methods

In the following code chunks, obj is considered to be an object of class StackedTrack.

Exported in the name space:

stacking signature(GdObject="StackedTrack"): return the current stacking type.

   Usage:

   stacking(GdObject)

   Examples:

    stacking(obj)

stacking<- signature(GdObject="StackedTrack", value="character"): set the object's stacking type to one in c(hide, dense, squish, pack,full).

Usage:

stacking<-(GdObject, value)

Additional Arguments:

 value: replacement value.

Examples:

 stacking(obj) <- "squish"

Internal methods:

drawGD signature(GdObject="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.

Usage:

drawGD(GdObject, minBase, maxBase, prepare=FALSE,subset=TRUE, ...)

Additional Arguments:

 minBase, maxBase: the coordinate range to plot.

 prepare: run method in preparation or in production mode.

 subset: subset the object to the visible region or skip the potentially expensive subsetting operation.

 ...: all further arguments are ignored.

Examples:

 Gviz:::drawGD(obj, prepare=FALSE)

setStacks signature(GdObject="StackedTrack"): recompute the stacks based on the available space and on the object's track items and stacking settings.

Usage:

setStacks(GdObject, from, to)

Additional Arguments:

 from, to: compute stacking within a certain coordinates range. This needs to be supplied for the plotting function to know the current genomic coordinates.

Examples:

 Gviz:::setStacks(obj)

stacks signature(GdObject="StackedTrack"): return the stack indices for each track item.

Usage:

stacks(GdObject)

Examples:

 Gviz:::stacks(obj)

initialize signature(.Object="StackedTrack"): initialize the object.

Inherited methods:

[signature(x="StackedTrack", i="ANY", j="ANY",drop="ANY"): subset the items in the StackedTrack object. This is essentially similar to subsetting of the GRanges object in the range slot. For most applications, the subset method may be more appropriate.

Additional Arguments:

　i, j: subsetting indices, j is ignored.

　drop: argument is ignored.

Examples:

　obj[1:5]

chromosome signature(GdObject="StackedTrack"): return the chromosome for which the track is defined.

Usage:

chromosome(GdObject)

Examples:

　chromosome(obj)

chromosome<- signature(GdObject="StackedTrack"): 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.

Usage:

chromosome<-(GdObject, value)

Additional Arguments:

　value: replacement value.

Examples:

　chromosome(obj) <- "chr12"

start, end, width signature(x="StackedTrack"): the start or end coordinates of the track items, or their width in genomic coordinates.

Usage:

start(x)

end(x)

width(x)

Examples:

　start(obj)

　end(obj)

　width(obj)

start<-, end<-, width<- signature(x="StackedTrack"): replace the start or end coordinates of the track items, or their width.

Usage:

start<-(x, value)

end<-(x, value)

width<-(x, value)

Additional Arguments:

　value: replacement value.

Examples:

　start(obj) <- 1:10

　end(obj) <- 20:30

width(obj) <- 1

position signature(GdObject="StackedTrack"): the arithmetic mean of the track item's coordionates, i.e., (end(obj)-start(obj))/2.

Usage:

position(GdObject)

Examples:

position(obj)

feature signature(GdObject="StackedTrack"): return the grouping information for track items. For certain sub-classes, groups may be indicated by different color schemes when plotting. See grouping or AnnotationTrack and GeneRegionTrack for details.

Usage:

feature(GdObject)

Examples:

feature(obj)

feature<- signature(gdObject="StackedTrack", value="character"): 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 StackedTrack. See grouping or AnnotationTrack and GeneRegionTrack for details.

Usage:

feature<-(GdObject, value)

Additional Arguments:

value: replacement value.

Examples:

feature(obj) <- c("a", "a", "b", "c", "a")

genome signature(x="StackedTrack"): return the track's genome.

Usage:

genome(x)

Examples:

genome(obj)

genome<- signature(x="StackedTrack"): set the track's genome. Usually this has to be a valid UCSC identifier, however this is not formally enforced here.

Usage:

genome<-(x, value)

Additional Arguments:

value: replacement value.

Examples:

genome(obj) <- "mm9"

length signature(x="StackedTrack"): return the number of items in the track.

Usage:

length(x)

Examples:

length(obj)

range signature(x="StackedTrack"): return the genomic coordinates for the track as an object of class IRanges.

Usage:

range(x)

Examples:

  range(obj)

ranges signature(x="StackedTrack"): return the genomic coordinates for the track along with all additional annotation information as an object of class GRanges.

Usage:

ranges(x)

Examples:

  ranges(obj)

split signature(x="StackedTrack"): split a StackedTrack 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 StackedTrack.

Usage:

split(x, f, ...)

Additional Arguments:

  f: the splitting factor.

  ...: all further arguments are ignored.

Examples:

  split(obj, c("a", "a", "b", "c", "a"))

strand signature(x="StackedTrack"): 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.

Usage:

strand(x)

Examples:

  strand(obj)

strand<- signature(x="StackedTrack"): replace the strand information for the track items. The replacement value needs to be an appropriate scalar or vector of strand values.

Usage:

strand<-(x, value)

Additional Arguments:

  value: replacement value.

Examples:

  strand(obj) <- "+"

values signature(x="StackedTrack"): return all additional annotation information except for the genomic coordinates for the track items as a data.frame.

Usage:

values(x)

Examples:

  values(obj)

coerce signature(from="StackedTrack",to="data.frame"): coerce the GRanges object in the range slot into a regular data.frame.

Examples:

  as(obj, "data.frame")

subset signature(x="StackedTrack"): subset a StackedTrack by coordinates and sort if necessary.

Usage:

subset(x, from, to, sort=FALSE, ...)

Additional Arguments:

  from, to: the coordinates range to subset to.

  sort: sort the object after subsetting. Usually not necessary.

  ...: additional arguments are ignored.

Examples:

  subset(obj, from=10, to=20, sort=TRUE)

displayPars signature(x="StackedTrack", name="character"): list the value of the display parameter name. See settings for details on display parameters and customization.

Usage:

displayPars(x, name)

Examples:

  displayPars(obj, "col")

displayPars signature(x="StackedTrack", name="missing"): list the value of all available display parameters. See settings for details on display parameters and customization.

Examples:

  displayPars(obj)

getPar signature(x="StackedTrack", name="character"): alias for the displayPars method. See settings for details on display parameters and customization.

Usage:

getPar(x, name)

Examples:

  getPar(obj, "col")

getPar signature(x="StackedTrack", name="missing"): alias for the displayPars method. See settings for details on display parameters and customization.

Examples:

  getPar(obj)

displayPars<- signature(x="StackedTrack", value="list"): set display parameters using the values of the named list in value. See settings for details on display parameters and customization.

Usage:

displayPars<-(x, value)

Examples:

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

setPar signature(x="StackedTrack", value="character"): set the single display parameter name to value. Note that display parameters in the StackedTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Usage:

setPar(x, name, value)

Additional Arguments:

  name: the name of the display parameter to set.

Examples:

  setPar(obj, "col", "red")

setPar signature(x="StackedTrack", value="list"): set display parameters by the values of the named list in value. Note that display parameters in the StackedTrack class are pass-by-reference, so no re-assignmnet to the symbol obj is necessary. See settings for details on display parameters and customization.

Examples:

  setPar(obj, list(col="red", lwd=2))

group signature(GdObject="StackedTrack"): return grouping information for the individual items in the track. Unless overwritten in one of the sub-classes, this usualy returns NULL.

Usage:

group(GdObject)

Examples:

  group(obj)

names signature(x="StackedTrack"): return the value of the name slot.

Usage:

names(x)

Examples:

  names(obj)

names<- signature(x="StackedTrack", value="character"): set the value of the name slot.

Usage:

names<-(x, value)

Examples:

  names(obj) <- "foo"

coords signature(ImageMap="StackedTrack"): return the coordinates from the internal image map.

Usage:

coords(ImageMap)

Examples:

  coords(obj)

tags signature(x="StackedTrack"): return the tags from the internal image map.

Usage:

tags(x)

Examples:

  tags(obj)

Display Parameters

The following display parameters are set for objects of class AnnotationTrack 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.

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.

Additional display parameters are being inherited from the respective parent classes. Note that not all of them may have an effect on the plotting of StackedTrack objects.

GdObject:

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

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 panels.

cex=1: Numeric scalar. The overall font expansion factor for all text.

cex.axis=NULL: Numeric scalar. The expansion factor for the axis annotation. Defaults to NULL, in which case it is computed 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="#0080FF": Integer or character scalar. Default line color 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 color for the y axis, if any.

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

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

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

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

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

collapse=TRUE: Boolean controlling wether 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.

fontcolor="black": Integer or character scalar. The font color for all text.

fontface=1: Integer or character scalar. The font face for all text.

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.

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

fontsize=12: Numeric scalar. The font size for all text.

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.

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

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

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

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

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.

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.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DisplayPars

GdObject

GeneRegionTrack

GRanges

ImageMap

IRanges

RangeTrack

collapsing

DataTrack

grouping

panel.grid

plotTracks

settings

UcscTrack	*Meta-constructor for GenomeGraph 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 ameanable to plotting with plotTracks.

clearSessionCache is can be called to remove all cached items from the session which are generated when connecting with the UCSC data base.

Usage

UcscTrack(track, table=NULL, trackType=c("AnnotationTrack",
"GeneRegionTrack", "DataTrack", "GenomeAxisTrack"), genome, chromosome,
name=NULL, from, to, ...)

clearSessionCache()

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", "GenomeAxisTr 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

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 dowloaded UCSC tables if available. For instance, assuming the table for track 'foo' contains the columns 'id', 'type', 'fromLoc' and 'toLoc', giving the featuer 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="AnnotationTr

To reduce the bandwidth, some caching of the UCSC connection takes place. In order to remove these cached session items, call clearSessionCache.

Value

An annotation track object as determined by trackType.

Author(s)

Florian Hahne

See Also

AnnotationTrack

DataTrack

GeneRegionTrack

GenomeAxisTrack

plotTracks

Index