Package 'Rgb'

February 8, 2015

Type Package

Title Rgb, the R Genome Browser

Version 1.4.0

Date 2015-02-08

Author Sylvain Mareschal

Maintainer Sylvain Mareschal <maressyl@gmail.com>

URL http://bioinformatics.ovsa.fr/Rgb

BugReports https://github.com/maressyl/R.Rgb

Description This package provides reference classes to efficiently handle (slice, annotate, draw ...) genomic features (such as genes or transcripts), and an interactive interface to browse them.

License file LICENSE

Suggests tcltk, tkrplot, tools

Depends R (>= 3.0.0), methods

R topics documented:

`Annotation` *Annotation track constructors*

Description

These functions constructs `track.table` inheriting objects from free annotation files.

Usage

```
track.table.GTF(file, name = NA, attr = "split", features = "exon", quiet = FALSE, ...)
track.exons.CCDS(file, name = "CCDS exons", ...)
track.CNV.DGV(file, name = "DGV CNV", ...)
track.genes.NCBI(file, name = "NCBI genes", selection, ...)
track.bands.UCSC(file, name = "UCSC bands", ...)
```

Arguments

`file`	Single character value, the path to the raw file to parse. See the 'Details' section below.
`name`	Single character value, the `name` field for the `track.table` object. For `track.table.GTF`, use `NA` to refer to the "source" column content (if it contains a unique value for all rows).
`attr`	To be passed to `read.gtf`.
`features`	To be passed to `read.gtf`.
`quiet`	To be passed to `read.gtf`.
`...`	Further arguments are passed to the class constructor, as a result most of the handled arguments are `track.table` arguments. Consider notably `.organism` and `.assembly` for track annotation.
`selection`	Character vector, filter to apply on the "group_label" column for NCBI genes. Raises an error with the possible values when missing.

Details

track.table.GTF imports a "Gene Feature Transfert" file, as proposed by the UCSC Table Browser at http://www.genome.ucsc.edu/cgi-bin/hgTables) for a large amount of species. See the read.gtf manual for further details.

track.exons.CCDS contains various transcripts from the "Consensus Coding DNA Sequence" project, currently only available for mouse and human (see the NCBI data repository at http://ftp.ncbi.nlm.nih.gov/pub/CCDS/, and look for a file named "CCDS_current.txt").

track.CNV.DGV parses constitutive copy number variations from the current version of the Database of Genomic Variants, downloadable from http://dgv.tcag.ca/dgv/app/downloads using "DGV Variants" links. The whole database is dedicated to the human specy only.

track.genes.NCBI parses the gene list from the MapView project of the NCBI, for one of many species available at http://ftp.ncbi.nih.gov/genomes/MapView/. Select your specy of interest, then browse "sequence", "current" and "initial_release" (if the directories are available, they are not for certain species). Download the file named "seq_gene.md.gz". As many assemblies are included in the file, a first call to the function without "selection" is required, to list the available values. A second call with the appropriate assembly name will produce the desired track file.

track.bands.UCSC produces a track of cytogenetic banding, as made available by the UCSC for many species at http://hgdownload.cse.ucsc.edu/downloads.html. Select the specy and assembly version that suits your needs, and look for a file named "cytoBand.txt.gz" in the "Annotation database" section.

Value

Return a track.table-inheriting object (of class track.exons, track.CNV, track.genes or track.bands).

Author(s)

Sylvain Mareschal

References

Example of track.exons.CCDS raw file (current human assembly) : http://ftp.ncbi.nlm.nih.gov/pub/CCDS/current_human/CCDS.current.txt

Example of track.CNV.DGV raw file (human assembly 'hg19') : http://dgv.tcag.ca/dgv/docs/GRCh37_hg19_variants_2013-05-31.txt

Example of track.genes.NCBI raw file (current human assembly) : http://ftp.ncbi.nih.gov/genomes/MapView/Homo_sapiens/sequence/current/initial_release/seq_gene.md.gz

Example of track.bands.UCSC raw file (human assembly 'hg19') : http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/cytoBand.txt.gz

See Also

track.fasta-constructors, Annotation

track.table-class, track.exons-class, track.CNV-class, track.genes-class or track.bands-class

tk.browse, browsePlot

Examples

```
# From the "How-to" vignette, section "Custom annotation tracks"
file <- system.file("extdata/Cosmic_ATM.gtf.gz", package="Rgb")
tt <- track.table.GTF(file)
```

`crossable-class` *Class* `"crossable"`

Description

Reference classes extending this virtual class must have a `slice` method, as a generic `cross` method based on it is provided.

Its only purpose is currently to add the `cross` method to `"track.table"`, as `"sliceable"` does not guarantee that `slice` returns a data.frame as `crossable` needs one.

Extends

Class `sliceable`, directly.
Class `drawable`, by class `sliceable`, distance 2.

All reference classes extend and inherit methods from `envRefClass`.

Fields

The following fields are inherited (from the corresponding class):

- name (drawable)
- parameters (drawable)

Methods

`cross(annotation, colname = , type = , fuzzyness = , maxElements = , location = , precision = ,`
Add a new column computed from overlaps with an other crossable object.
- **annotation** : other crossable object to compute overlap with.
- **colname** : single character value, the name of the new column to add to .self. If NULL or NA, the result will be returned rather than added to .self.
- **type** : single character value, either :
'cover', to compute coverage of 'annotation' elements for each .self element
'count', to count 'annotation' elements overlapping each .self element
'cytoband', to get cytogenetic coordinates from a cytoband annotation track
an 'annotation' column name, to list 'annotation' elements overlapping each .self element
- **fuzzyness** : single integer value, to be added on each side of .self elements when computing overlaps.
- **maxElements** : single integer value, when more overlaps are found, lists are replaced by counts. Can be NA to disable this behavior.
- **location** : character vector, the 'chrom' / 'start' / 'end' .self columns to use for annotation.
- **precision** : single integer value from 1 to 4, amount of digits to consider in banding (type='cytoband').
- **quiet** : single logical value, wether to throw progression messages or not.

The following methods are inherited (from the corresponding class):

- callParams (drawable)
- callSuper (envRefClass)
- check (drawable)
- chromosomes (drawable)
- copy (envRefClass)

- defaultParams (sliceable)
- draw (sliceable)
- export (envRefClass)
- field (envRefClass)
- fix.param (drawable)
- getChromEnd (sliceable)
- getClass (envRefClass)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- import (envRefClass)
- initFields (envRefClass)
- initialize (drawable)
- setName (drawable)
- setParam (drawable)
- show (sliceable)
- slice (sliceable)
- trace (envRefClass)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

drawable, sliceable, track.table, cghRA.array

`draw.bg` *Background for track plots*

Description

This function draws the background for the other track ploting functions.

Usage

```
draw.bg(start, end, ylab = "", ysub = as.character(NA), mar = c(0.2, 5, 0.2, 1),
  xaxt = "s", yaxt = "n", yaxs = "r", ylim = c(0, 1), cex.lab = 1, bty = "o", ...)
```

Arguments

start	Single integer value, the left boundary of the window, in base pairs.
end	Single integer value, the right boundary of the window, in base pairs.
ylab	The name of the Y axis. See par.
ysub	Similar to ylab, but written on a closer line when yxat is "n". Can be NA to disable it.
mar	A numerical vector of the form "c(bottom, left, top, right)" which gives the number of lines of margin to be specified on the four sides of the plot. See par.
xaxt	Whether to plot an X axis ("s") or not ("n"). See par.
yaxt	Whether to plot an Y axis ("s") or not ("n"). If no Y axis is drawn, ysub may be used to plot a sub-title to the Y axis. See par.
yaxs	Y axis style, "r" enlarges the Y limits by 4 percents on each side for a cleaner look, "i" will not. See par.
ylim	The Y axis limits as a numerical vector of the form "c(start, end)" of the plot. Note that start > end is allowed and leads to a "reversed axis". Use "NULL" to guess the axis range from the data. See plot.default.
cex.lab	The relative character size of x and y axis labels (default: 1). See par.
bty	A character string which determined the type of box which is drawn about plots. If bty is one of "o" (the default), "l", "7", "c", "u", or "]" the resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box. See par.
...	Not used, only ignores other arguments.

Author(s)

Sylvain Mareschal

See Also

draw.boxes, draw.hist, draw.points, draw.pileup, draw.seq

---

draw.boxes *Boxes plot of a track*

Description

This function draws a slice of a track content, with a distinct box for each track element.

Usage

```
draw.boxes(slice, start, end, maxElements = 50, maxDepth = 100, label = TRUE,
  labelStrand = FALSE, labelCex = 0.8, labelSrt = 0, labelAdj = "center",
  labelOverflow = TRUE, labelFamily = "sans", colorVal = "#BBBBBB",
  colorFun = function() NULL, border = "#666666", cex.lab = 1, spacing = 0.2,
  bty = "o", groupBy = NA, groupPosition = NA, groupSize = NA, ...)
```

Arguments

slice	A data.frame holding the data to plot, with elements in rows and data in columns.
start	Single integer value, the left boundary of the window, in base pairs.
end	Single integer value, the right boundary of the window, in base pairs.
maxElements	Single integer value, the maximum amount of boxes on the plot (if exhausted, only the amount of elements will be ploted).
maxDepth	Single integer value, the maximum amount of box heights allowed on the plot to avoid overlaps (if exhausted an error message will be ploted, turning label to FALSE might help).
label	Single logical value, whether to print labels on boxes or not.
labelStrand	Single logical value, whether to add the strand at the end of labels or not.
labelCex	Single numeric value, character expansion factor for labels.
labelSrt	Single numeric value, string rotation angle for labels.
labelAdj	'left', 'right' or 'center', the horizontal adjustement of the labels on the boxes.
labelOverflow	Single logical value, whether to write labels on boxes too narrow to host them or not.
labelFamily	Single character value, the font family to use for labels ('serif', 'sans', 'mono' or 'Hershey'). 'serif' and 'sans' are not monospaced fonts, so label box sizes and collision handling might not work as expected with them.
colorVal	The color to fill boxes with (as a name, an integer or an hexadecimal character description).
colorFun	A function with no arguments, which returns a vector of as many colors than the slice has rows. It can make use of any argument described on this page (including custom arguments passed via "..."), as its enclosing environment is redefined to the calling one. colorVal must be NA for the function to be used.
border	The color to fill box borders with (as a name, an integer or an hexadecimal character description).
cex.lab	The relative character size of x and y axis labels (default: 1). See par.
spacing	Single numeric value, the vertical spacing between boxes, in proportion of the box height.
bty	A character string which determined the type of box which is drawn about plots. If bty is one of "o" (the default), "l", "7", "c", "u", or "]" the resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box. See par.
groupBy	Single character value, the name of a slice column to use for feature grouping. Grouped features are drawn on the same line, and joined by an horizontal segment. Use NA to disable feature grouping.
groupPosition	Single character value, the name of an integer slice column with the position of the feature in the group (starting at 1). If groupBy is provided with groupPosition and groupSize, groups containing out-of-range elements will be extended to the screen boundary. Use NA if feature grouping is disabled, or if awareness of out-of-range features is not critical.
groupSize	Single character value, the name of an integer slice column with the total amount of features in the group. See groupPosition.
...	Further arguments to be passed to draw.bg.

Author(s)

Sylvain Mareschal

See Also

draw.bg, draw.hist, draw.points, draw.pileup, draw.seq

---

draw.hist *Histogram plot of a track*

Description

This function draws a slice of a track content, with a distinct vertical bar for each track element.

Usage

```
draw.hist(slice, start, end, column = "value", colorVal = "#666666",
  colorFun = function() NULL, border = "#666666", cex.lab = 1, origin = 0,
  bty = "o", ...)
```

Arguments

slice	A data.frame holding the data to plot, with elements in rows and data in columns.
start	Single integer value, the left boundary of the window, in base pairs.
end	Single integer value, the right boundary of the window, in base pairs.
column	Single character value, the name of the slice column to use for bar heights.
colorVal	The color to fill bars with (as a name, an integer or an hexadecimal character description).
colorFun	A function with no arguments, which returns a vector of as many colors than the slice has rows. It can make use of any argument described on this page (including custom arguments passed via "..."), as its enclosing environment is redefined to the calling one. colorVal must be NA for the function to be used.
border	The color to fill boxe borders with (as a name, an integer or an hexadecimal character description).
cex.lab	The relative character size of x and y axis labels (default: 1). See par.
origin	Single numeric value, the Y value of the horizontal side common to all boxes.
bty	A character string which determined the type of box which is drawn about plots. If bty is one of "o" (the default), "l", "7", "c", "u", or "]" the resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box. See par.
...	Further arguments to be passed to draw.bg.

Author(s)

Sylvain Mareschal

See Also

draw.bg, draw.boxes, draw.points, draw.pileup, draw.seq

draw.pileup *Pileup plot of a BAM track*

Description

This function draws a slice of a sequence pileup, highlighting polymorphisms.

Usage

```
draw.pileup(slice, start, end, ylim = NA, bty = "o", label = TRUE, labelCex = 0.75,
  bases = c(A = "#44CC44", C = "#4444CC", G = "#FFCC00", T = "#CC4444"),
  maxRange = 500, cex.lab = 1, alphaOrder = 3, alphaMin = 0.1, ...)
```

Arguments

slice	An integer matrix of read counts, with nucleotides in rows and positions in columns. Both dimensions must be named.
start	Single integer value, the left boundary of the window, in base pairs.
end	Single integer value, the right boundary of the window, in base pairs.
ylim	See plot.default. NA will compute a dynamic ylim fiting the data.
bty	A character string which determined the type of box which is drawn about plots. If bty is one of "o" (the default), "l", "7", "c", "u", or "]" the resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box. See par.
label	Single logical value, whether to print nucleotide on bars or not.
labelCex	Single numeric value, character expansion factor for labels.
bases	Named character vector, defining the color to use for each nucleotide.
maxRange	Single integer value, nothing will be ploted if the plot window is wider by this value (in bases).
cex.lab	The relative character size of x and y axis labels (default: 1). See par.
alphaOrder	Single numeric value, the order of the formula controlling the transparency. Increase this value to increase sensitivity to rare variants.
alphaMin	Single numeric value, the minimal intensity in the formula controlling the transparency (between 0 and 1). Perfectly homozyguous positions will typically use this intensity of color.
...	Further arguments to be passed to draw.bg.

Author(s)

Sylvain Mareschal

See Also

track.bam-class

draw.bg, draw.boxes, draw.hist, draw.points, draw.seq

`draw.points` *Scatter plot of a track*

Description

This function draws a slice of a track content, with a distinct point for each track element.

Usage

```
draw.points(slice, start, end, column = "value", colorVal = "#666666",
  colorFun = function() NULL, cex.lab = 1, cex = 0.6, pch = "+", bty = "o", ...)
```

Arguments

`slice`	A `data.frame` holding the data to plot, with elements in rows and data in columns.
`start`	Single integer value, the left boundary of the window, in base pairs.
`end`	Single integer value, the right boundary of the window, in base pairs.
`column`	Single character value, the name of the `slice` column to use for bar heights.
`colorVal`	The color to fill bars with (as a name, an integer or an hexadecimal character description).
`colorFun`	A function with no arguments, which returns a vector of as many colors than the `slice` has rows. It can make use of any argument described on this page (including custom arguments passed via "..."), as its enclosing environment is redefined to the calling one. `colorVal` must be `NA` for the function to be used.
`cex.lab`	See `par`.
`cex`	See `par`.
`pch`	See `par`.
`bty`	See `par`.
`...`	Further arguments to be passed to `draw.bg`.

Author(s)

Sylvain Mareschal

See Also

`draw.bg`, `draw.boxes`, `draw.hist`, `draw.pileup`, `draw.seq`

`draw.seq` *Plot a sequence of nucleotides*

Description

This function draws a slice of a character vector, with labels and distinct colors for each nucleotide.

Usage

```
draw.seq(slice = NULL, start, end, bty = "o", labelCex = 0.75,
  bases = c(A = "#44CC44", C = "#4444CC", G = "#FFCC00", T = "#CC4444"),
  maxRange = 500, cex.lab = 1, ...)
```

Arguments

`slice`	Character vector, with a single letter per element.
`start`	Single integer value, the left boundary of the window, in base pairs.
`end`	Single integer value, the right boundary of the window, in base pairs.
`bty`	A character string which determined the type of box which is drawn about plots. If bty is one of "o" (the default), "l", "7", "c", "u", or "]" the resulting box resembles the corresponding upper case letter. A value of "n" suppresses the box. See `par`.
`labelCex`	Single numeric value, character expansion factor for labels.
`bases`	Named character vector, defining the color to use for each nucleotide (names have to be uppercase, `slice` is converted to uppercase for matching).
`maxRange`	Single integer value, nothing will be ploted if the plot window is wider by this value (in bases).
`cex.lab`	The relative character size of x and y axis labels (default: 1). See `par`.
`...`	Further arguments to be passed to `draw.bg`.

Author(s)

Sylvain Mareschal

See Also

`track.fasta-class`

`draw.bg`, `draw.boxes`, `draw.hist`, `draw.points`, `draw.pileup`

drawable-class *Class* "drawable"

Description

Reference classes extending this virtual class must have a draw method, so their objects can be managed by tk.browse and browsePlot.

Extends

All reference classes extend and inherit methods from envRefClass.

Fields

name: Custom name for the object, as a character vector of length 1.

parameters: A list, storing object-specific parameters to use as draw arguments.

Methods

callParams(chrom, start, end, ...): Called with draw() arguments, it returns the final argument list handling default and overloaded parameters.
- **chrom, start, end, ...** : arguments passed to draw().

check(warn =): Raises an error if the object is not valid, else returns TRUE

chromosomes(): [Virtual method]
Returns the chromosome list as a vector. NULL is valid if non relevant, but should be avoided when possible.

defaultParams(...): Returns class-specific defaults for graphical parameters. Inheriting class should overload it to define their own defaults.
- **...** : may be used by inheriting methods, especially for inter-dependant parameters.

draw(chrom, start = , end = , ...): [Virtual method]
Draws the object content corresponding to the defined genomic window, usually in a single plot area with coordinates in x and custom data in y.
Overloading methods should use .self$callParams(chrom, start, end ...) to handle drawing parameters and NA coordinates in a consistent way.
- **chrom** : single integer, numeric or character value, the chromosomal location.
- **start** : single integer or numeric value, inferior boundary of the window. NA should refer to 0.
- **end** : single integer or numeric value, superior boundary of the window. NA should refer to .self$getChromEnd().
- **...** : additionnal drawing parameters (precede but do not overwrite parameters stored in the object).

fix.param(parent =): Edit drawing parameters using a Tcl-tk GUI
- **parent** : tcltk parent frame for inclusion, or NULL.

getChromEnd(chrom): [Virtual method]
Returns as a single integer value the ending position of the object description of the given chromosome. NA (integer) is valid if non relevant, but should be avoided when possible.
- **chrom** : single integer, numeric or character value, the chromosomal location. NA is not required to be handled.

getName(): 'name' field accessor.

`getParam(name, ...)`: Returns the parameter stored, or the default value if no custom value is stored for it.
- **name** : single character value, the name of the parameter to return.
- **...** : to be passed to defaultParams(), especially for inter-dependant parameters.

`initialize(name = , parameters = , ...)`:

`setName(value)`: 'name' field mutator.

`setParam(name, value)`: Updates a parameter stored in the object.
- **name** : single character value, the name of the parameter to set.
- **value** : the new value to assign to the parameter (any type). If missing the parameter is discarded, thus returning to dynamic default value.

The following methods are inherited (from the corresponding class):

- callSuper (envRefClass)
- copy (envRefClass)
- export (envRefClass)
- field (envRefClass)
- getClass (envRefClass)
- getRefClass (envRefClass)
- import (envRefClass)
- initFields (envRefClass)
- show (envRefClass, overloaded)
- trace (envRefClass)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

`sliceable`, `crossable`

`drawable.list` *Drawable list constructor*

Description

Produces a `drawable.list` object, for `tk.browse` and `browsePlot` input.

Usage

```
drawable.list(files = character(0), objects = NULL, hidden = FALSE, warn = TRUE)
```

Arguments

files	Character vector, path and names of the files holding the drawables object (with .rdt or .rds extensions, see saveRDT and saveRDS for further details). Can contain NA values, but providing a path allows updates to be saved.
objects	List of drawable-inheriting objects to include in the list. If NULL, objects will be extracted from files, else a list with as many elements as files is required.
hidden	Logical vector, whether to show tracks in tk.browse and browsePlot or not (this choice is not definitive). Recycled if necessary.
warn	Single logical value, to be passed to the appropriate check method.

Value

A drawable.list object.

Author(s)

Sylvain Mareschal

See Also

drawable.list-class, drawable-class, tk.browse, browsePlot

---

drawable.list-class *Class* "drawable.list"

Description

The purpose of this class is to store and manage a collection of drawable objects. These collections are to be used by tk.browse and browsePlot as input.

Objects can be created by the drawable.list constructor, and edited / created using the tk.tracks Tcl-tk interface.

Extends

All reference classes extend and inherit methods from envRefClass.

Fields

classes: Read-only, returns a vector of objects classes.

count: Read-only, returns the length of objects, as a single integer.

files: Character vector, the paths where each drawable object is to be stored.

hidden: Logical vector, whether each object is to drawn or hidden in plots.

names: Read-only, returns a vector of objects 'name' fields.

objects: List of drawable-inheriting objects.

Methods

`add(file, track = , hidden = ):` Add a track to the list.
- **file** : single character value, the path to the file containing the 'drawable' object to add.
- **track** : a 'drawable' object to add. If NULL, will be extracted from 'file'.
- **hidden** : single logical value, whether the track is to be shown on plots or hidden. This value can be changed later.

`check(warn = ):` Raises an error if the object is not valid, else returns TRUE

`fix.files(parent = ):` Edit drawable list using a Tcl-tk GUI
- **parent** : tcltk parent frame for inclusion, or NULL.

`fix.param(selection = , parent = ):` Edit drawing parameters using a Tcl-tk GUI
- **selection** : single integer value, the position of the track selected in the list.
- **parent** : tcltk parent frame for inclusion, or NULL.

`get(index, what = ):` Returns a single 'what' from the series
- **index** : single numeric value, the position of the track to get.
- **what** : single character value, the field to be exracted.

`getByClasses(classes, what = ):` Returns a subset of 'what' from the series, querying by class inheritance
- **classes** : character vector, the class names of the objects to get (inheriting classes are picked too).
- **what** : single character value, the field to be exracted.

`getByNames(names, what = ):` Returns a subset of 'what' from the series, querying by track name
- **names** : character vector, the names of the objects to get.
- **what** : single character value, the field to be exracted.

`getByPositions(positions, what = ):` Returns a subset of 'what' from the series, querying by position
- **positions** : integer vector, the positions of the objects to get.
- **what** : single character value, the field to be exracted.

`initialize(files = , objects = , hidden = , ...):`

`moveDown(toMove):` Increases the position of a track, switching position with the next one
- **toMove** : single numeric value, the position of the track to move.

`moveUp(toMove):` Decreases the position of a track, switching position with the previous one
- **toMove** : single numeric value, the position of the track to move.

`remove(toRemove):` Remove one or many tracks from the list
- **toRemove** : numeric vector, the positions of the tracks to remove.

The following methods are inherited (from the corresponding class):

- callSuper (envRefClass)
- copy (envRefClass)
- export (envRefClass)
- field (envRefClass)
- getClass (envRefClass)
- getRefClass (envRefClass)
- import (envRefClass)
- initFields (envRefClass)
- show (envRefClass, overloaded)

- trace (envRefClass)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

drawable.list, drawable-class, tk.browse, browsePlot

---

findDrawables *Find drawable objects in memory*

Description

This function searches an environment for drawable-class inheriting objects.

Usage

```
findDrawables(varNames = NA, envir = globalenv())
```

Arguments

varNames	Character vector, the R expression(s) of potential drawable-class inheriting objects (or lists of such objects) to check. If NA, an object name list will be generated from envir. Non-NA values are mainly used for internal recursive calls on lists, users should not have to change it.
envir	The environment to look into for drawable-class inheriting objects.

Details

Objects are currently found if defined as individual variables, as parts of drawable.list objects or into standard R lists. lists are explored recursively, so lists embedded into other lists are explored too, whatever their depths.

Value

Returns a character vector containing the R expression(s) to be evaluated in envir to get the drawable-class inheriting objects.

This vector carries an "envir" attribute containing the value passed to this function via the envir argument.

Author(s)

Sylvain Mareschal

See Also

tk.tracks

hsFeatures *Homo sapiens GRCh37 genomic features*

Description

hsGenes contains 8000 randomly choosen human genes, from the NCBI repository.

hsBands contains the whole list of human chromosome G-banding, from the UCSC repository.

Both are provided as exemplar data of genomic features.

Usage

```
hsGenes
hsBands
```

Format

For hsGenes, a data.frame with 8000 rows and the following columns : "chrom", "start", "end" and "name".

For hsBands, a track.table object with 862 rows and the following columns : "name", "chrom", "strand", "start", "end", "stain".

Source

National Center for Biotechnology Information (ftp.ncbi.nih.gov)

University of California, Santa Cruz (genome.ucsc.edu)

RDT storage files *Single refTable object storage*

Description

Functions to write a single refTable object to a file, and to restore it.

Usage

```
saveRDT(object, file, compress = "gzip", compression_level = 6)
readRDT(file, version = FALSE)
```

Arguments

object	An object of class refTable to store.
file	A connection or the name of the file where the R object is saved to or read from. The '.rdt' file extension is recommended, but not mandatory.
compress	To be passed to save
compression_level	To be passed to save
version	Single logical value, whether to return the stored object or the version of the package used to store it.

Details

These functions mimic the saveRDS and saveRDS system, without storing the class definition in the file (which can lead to about 100 useless Ko of data and longer loading times). It is intented to manage all classes extending refTable, but no guarantee is provided for classes with non-atomic slots (particularly environment-derived ones).

Value

saveRDT returns nothing, readRDT returns the object stored in the file or a single character value (depends on the version argument).

Note

To avoid whole-environment copying, environments of function slots are discarded.

Author(s)

Sylvain Mareschal

See Also

refTable-class, saveRDS, saveRDS

---

read.gtf *Parses a GTF2 file*

Description

This function parses a simple "Gene Transfer Format" (GTF2.2) into a data.frame, as distributed by the UCSC Table Browser.

As this format is an extension of the "Gene Feature Format" (GFF3), some retro-compatibility can be expected but not guaranteed.

Usage

```
read.gtf(file, attr = c("split", "intact", "skip"), features = NULL, quiet = FALSE)
```

Arguments

file	Single character value, the path and name of the GTF2 file to parse (possibly gzipped).
attr	Single character value, defining how to deal with attributes. "skip" discards the attributes data, "intact" does not process it and "split" adds a column for each attribute (identified by their names).
features	Character vector, if not NULL only rows with a "feature" column value from this list will be kept.
quiet	Single logical value, whether to send diagnostic messages or not.

Value

A `data.frame` with the standard GTF2 columns. The "strand" column is converted to `factor`, "?" are turned to `NA` and "." are kept for features where stranding is not relevant (See the GFF3 specification).

Note

Currently not implemented :

- FASTA section and sequences (error raising)
- Special character escaping (error raising)
- Attribute quotes (kept)
- Sections (all data pooled)
- Meta data (ignored)

Author(s)

Sylvain Mareschal

References

GTF2.2 specification : `http://mblab.wustl.edu/GTF22.html`

GFF3 Sequence Ontology specification : `http://www.sequenceontology.org/gff3.shtml`

See Also

`track.table.GTF`

`refTable-class` *Class* "`refTable`"

Description

This class is similar to the `data.frame` standard R class, following the object-oriented paradigm. The use of Reference Class system allows significative memory and time saving, making this class more suitable than `data.frame` to handle large tabular data.

Objects can be created by two distincts means :

- The `refTable` constructor, similar to the `data.frame` constructor. It imports a single `data.frame` or a collection of vectors into the object, and check immediatly for validity.
- The `new` function (standard behavior for S4 and reference classes), which produces an empty object and do NOT check for validity. You can provide as arguments the values to use as the new object fields, if you know what you are doing.

Extends

All reference classes extend and inherit methods from `envRefClass`.

Implementation

Data storage: refTable objects store data into their `values` field as vectors. As `values` is an environment, manipulating a refTable implies a pass-by-reference paradigm rather than the standard R pass-by-copy, i.e. data duplication (and so time and memory wasting) is widely reduced. As an example, updating a single cell in a `data.frame` leads to the duplication of the whole table in memory ('before' and 'after' versions), while in a refTable the duplication is limited to the involved column.

Column names: To facilitate column renaming, the vectors in `values` are not named according to the user-level column names, but according to references stored in the `colReferences` field (integers greater than 0 converted to characters). Rename a column only updates the `colNames` field and leave the `values` one alone, as the column reference does not change.

Data extraction: Data extracted from refTable are usually returned as `data.frame`, for a more comfortable R usage. The extraction mechanism handles `data.frame` extraction mechanisms, and relies on the `indexes` method to handle the others.

- Rows and columns may be selected by a numeric vector, as for R `data.frame` and vectors.
- They also may be selected by a logical vector, defining for each row / column if it is to be selected (`TRUE`) or not (`FALSE`). Such vectors are recycled if not long enough to cover all the rows / columns.
- A character vector defining the names of the rows / columns to select may also be used to extract data.
- The `NULL` value may be used to select all rows / columns.
- An unevaluated expression, as returned by `expression` or `parse` may be used to select rows in the table environment. See 'examples'.

Fields

`colCount`**:** Single `integer` value, the amount of rows in the table.

`colIterator`**:** Single `integer` value, last column reference used.

`colNames`**:** `Character` vector, the names of all rows (may be empty).

`colReferences`**:** `Character` vector, the column names in the `values` environment.

`rowCount`**:** Single `integer` value, the amount of rows in the table.

`rowNamed`**:** Single `logical` value, whether row names should be considered or not.

`rowNames`**:** `Character` vector, the names of all rows (may be empty).

`values`**:** An `environment` storing the columns as vectors.

Methods

`addColumn(content, name, after = )`**:** Adds a column in the table
- **content** : values to fill in the new column, as a vector.
- **name** : name of the new column, as character.
- **after** : where to add the column, as the index (numeric) or name (character) of the column on its left

`addDataFrame(dataFrame)`**:** Adds a data.frame content to the refTable
- **dataFrame** : the data to add.

`addEmptyRows(amount, newNames)`**:** Add rows filled with NA at the bottom of the table.
- **amount** : single integer value, the amount of rows to add.
- **newNames** : character vector, the names of the new rows. Ignored if the table is not row named.

`addList(dataList, row.names)`**:** Adds a list content to the refTable
- **dataList** : the data to add.
- **row.names** : character vector with the names of the enw rows.

`addVectors(..., row.names)`**:** Adds vectors to the refTable
- **...** : named vectors to add.

`check(warn = )`**:** Raises an error if the object is not valid, else returns TRUE

`coerce(j = , class, levels, ...)`**:** Coerces a single column to a different class
- **j** : column index (numeric) or name (character).
- **class** : name of the class to coerce 'j' to.
- **levels** : if 'class' is factor, the levels to use.
- **...** : further arguments to be passed to the 'as' method (for atomics) or function (for other classes).

`colOrder(newOrder, na.last = , decreasing = )`**:** Reorder the columns of the tables (duplication / subsetting are NOT handled)
- **newOrder** : new order to apply, as an integer vector of character vector of column names.

`delColumns(targets)`**:** Deletes a column from the table
- **targets** : character vector, the name(s) of the column(s) to delete.

`erase()`**:** Remove all the rows and columns in the table.

`extract(i = , j = , drop = , asObject = )`**:** Extracts values into a data.frame or vector
- **i** : row selection, see indexes() for further details.
- **j** : column selection, see indexes() for further details.
- **drop** : if TRUE and querying a single column, will return a vector instead of a data.frame.
- **asObject** : if TRUE results will be served in the same class as the current object.

`fill(i = , j = , newValues)`**:** Replaces values in a single column
- **i** : row indexes (numeric) or names (character). NULL or missing for all rows.
- **j** : column index (numeric) or name (character).
- **newValues** : vector of values to put in the object

`getColCount()`**:** 'colCount' field accessor.

`getColNames()`**:** 'colNames' field accessor.

`getLevels(j = )`**:** Get levels of a factor column
- **j** : column index (numeric) or name (character).

`getRowCount()`**:** 'rowCount' field accessor.

`getRowNames()`**:** 'rowNames' field accessor.

`indexes(i, type = )`**:** Checks row or column references and return numeric indexes
- **i** : reference to the rows or columns to select (NA not allowed), as :
- missing or NULL (all rows or columns)
- vector of numeric indexes to select
- vector of character indexes to select (if the dimension is named)
- vector of logical with TRUE on each value to select, FALSE otherwise
- expression object (as returned by e(...)), to be evaluated in the 'values' environment

`initialize(rowCount = , rowNames = , rowNamed = , colCount = , colNames = , colReferences = , co`

`metaFields()`**:** Returns a character vector of fields that do not directly depend on the tabular content, for clonage.

`rowOrder(newOrder, na.last = , decreasing = )`**:** Reorder the rows of the tables (duplication / subsetting are handled)
- **newOrder** : new order to apply, as an integer vector of row indexes or a character vector of

column names.
- **na.last** : to be passed to order(), if 'newOrder' is a column name vector.
- **decreasing** : to be passed to order(), if 'newOrder' is a column name vector.

`setColNames(j, value)`**:** Replaces one or many column names.
- **j** : subset of columns to rename.
- **value** : new column names to use, as a character vector.

`setLevels(j = , newLevels)`**:** Get or replace levels of a factor column
- **j** : column index (numeric) or name (character).
- **newLevels** : new levels to use, as a character vector.

`setRowNames(value)`**:** Replaces the entire row names set.
- **value** : new row names to use in the table, as a character vector. NULL will disable row naming.

`types(j = )`**:** Returns classes of selected columns
- **j** : column indexes or names (NULL for all columns)

The following methods are inherited (from the corresponding class):

- callSuper (envRefClass)
- copy (envRefClass, overloaded)
- export (envRefClass)
- field (envRefClass)
- getClass (envRefClass)
- getRefClass (envRefClass)
- import (envRefClass)
- initFields (envRefClass)
- show (envRefClass, overloaded)
- trace (envRefClass)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

Examples

```
# New empty refTable
tab <- new("refTable")
tab$addColumn(1:5, "C1")
tab$addColumn(letters[1:5], "C2")
tab$setRowNames(LETTERS[11:15])

# New filled refTable (same content)
tab <- refTable(C1=1:5, C2=letters[1:5], row.names=LETTERS[11:15])

# Whole table print
print(tab$extract())

# Data update
tab$fill(c(2,4), 2, c("B","D"))
```

```
# Data extraction
print(tab$extract(1:3))
print(tab$extract(c(TRUE, FALSE)))
print(tab$extract("K", "C1"))

# Expression-based extraction
expr <- expression(C1 %% 2 == 1)
print(tab$extract(expr))

# Table extension
tab$addEmptyRows(5L, LETTERS[1:5])
tab$fill(6:10, "C1", 6:10)
print(tab$extract())

# Filling from R objects
tab <- new("refTable")
print(tab$extract())
tab$addVectors(C1=1:5, C2=letters[1:5])
print(tab$extract())
tab$addList(list(C1=6:8, C3=LETTERS[6:8]))
print(tab$extract())

# Beware of recycling !
tab$addVectors(C1=9:15, C3=LETTERS[9:10])
print(tab$extract())
```

`refTable-constructor` *refTable class constructor*

Description

This function returns a new `refTable` object from various arguments.

Notice the `new()` alternative can be used to produce an empty object, setting only the fields not the content.

Usage

```
refTable(..., row.names, warn = TRUE)
```

Arguments

`...`	A `data.frame`, a `list` or a set of named vectors to use as columns. For list and vectors, `row.names` will be used as row names if provided.
`row.names`	Character vector, the names of the rows for list or vector input.
`warn`	Single logical value, to be passed to the `refTable` check method.

Value

An object of class `refTable`.

Author(s)

Sylvain Mareschal

See Also

refTable-class

Examples

```
# From vectors
tab <- refTable(colA=1:5, colB=letters[1:5])
print(tab$extract(3,))

# From list (recycling)
columns <- list(number=1, letters=LETTERS)
tab <- refTable(columns)
print(tab$extract())

# data.frame conversion
dataFrame <- data.frame(colA=1:5, colB=letters[1:5])
tab <- refTable(dataFrame)
print(tab$extract())
```

---

segMerge *Merges consecutive segments*

Description

Given a set of segments defined by "chrom", "start", "end" and various data, it merges consecutive rows (sorted by "chrom" then "start") that share same data. As an example, it is useful to merge consecutive regions of the genome sharing same copy numbers after modelization, or filling small gaps.

Usage

```
segMerge(segTable, on = names(segTable), fun = list(unique, start=min, end=max))
```

Arguments

segTable	A data.frame of segments, with at least "chrom", "start" and "end" columns. Standard behavior (default fun value) assumes "start" and "end" to be at least numeric, preferably integer.
on	Character vector, segTable columns that must all be identical for the consecutive rows to be merged. For convenience, "chrom" is forced in and "start" / "end" are forced out.
fun	A list of functions, defining how to merge values when merging rows. It should contain an unamed element for the default function, and named elements to deal with specific columns.

Value

Returns a data.frame similar to segTable.

Author(s)

Sylvain Mareschal

See Also

segOverlap

---

segOverlap *Merges overlapping segments*

Description

Given a set of segments defined by "chrom", "start", "end" and various data, it merges overlapping or jointive rows.

Usage

```
segOverlap(segTable, fun = list(unique, start=min, end=max))
```

Arguments

segTable	A data.frame of segments, with at least "chrom", "start" and "end" columns. Standard behavior (default fun value) assumes "start" and "end" to be at least numeric, preferably integer.
fun	A list of functions, defining how to merge values when merging rows. It should contain an unamed element for the default function, and named elements to deal with specific columns.

Value

Returns a data.frame similar to segTable.

Author(s)

Sylvain Mareschal

See Also

segMerge

---

singlePlot *Whole genome plot of a single track content*

Description

This function plots all chromosomes on a single plot, with boxes representing the track content.

Usage

```
singlePlot(track, bandTrack, columns = 4, exclude = c("X", "Y"))
```

Arguments

track	A track.table inheriting object, whose elements will be plotted. If height and colors column exist, they will be used to shape the boxes plotted.
bandTrack	A cytoband track, as returned by track.bands.UCSC.
columns	Single integer value, column count in the chromosome layout.
exclude	Character vector, the names of the chromosomes to not plot.

Author(s)

Sylvain Mareschal

See Also

track.table

---

sliceable-class *Class* "sliceable"

Description

Reference classes extending this virtual class must have a slice method, as a generic draw method based on it is provided.

Extends

Class drawable, directly.

All reference classes extend and inherit methods from envRefClass.

Fields

The following fields are inherited (from the corresponding class):

- name (drawable)
- parameters (drawable)

Methods

slice(chrom, start, end): [Virtual method]
Extract elements in the specified window, in a format suitable to draw().
- **chrom** : single integer, numeric or character value, the chromosomal location. NA is not handled.
- **start** : single integer or numeric value, inferior boundary of the window. NA is not handled.
- **end** : single integer or numeric value, superior boundary of the window. NA is not handled.

The following methods are inherited (from the corresponding class):

- callParams (drawable)
- callSuper (envRefClass)
- check (drawable)
- chromosomes (drawable)

- copy (envRefClass)
- defaultParams (drawable, overloaded)
- draw (drawable, overloaded)
- export (envRefClass)
- field (envRefClass)
- fix.param (drawable)
- getChromEnd (drawable, overloaded)
- getClass (envRefClass)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- import (envRefClass)
- initFields (envRefClass)
- initialize (drawable)
- setName (drawable)
- setParam (drawable)
- show (drawable, overloaded)
- trace (envRefClass)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

`drawable`, `crossable`, `cghRA.array`

`subtrack` *Extract elements within a genomic window*

Description

`subtrack` extracts lines from a `data.frame`, `list` or `vector` collection within a single genomic window, defined by a chromosome name, a starting and an ending positions. As this is a common task in genome-wide analysis, this function relies on an optimized C code in order to achieve good performances.

`sizetrack` is very similar to `subtrack`, but only count lines without extracting the data.

`istrack` checks if a collection of data is suitable for `subtrack` and `sizetrack` (See 'Track definition' for further details). As this operation is quite expensive and should be performed once, it is up to the user to check its data before subtracking.

Usage

```
istrack(...)
subtrack(...)
sizetrack(...)
```

Arguments

...	A collection of data to be considered as a single track. Named vectors are considered as single columns, `data.frame` and `list` as collections of columns, all parallelized in a single bidimensional table (assuming they all have same lengths / row counts). See 'Track definition' for further details. For `subtrack` and `sizetrack`, the first arguments must be the following (preferably unnamed) : • chromosome location (`integer` or `character`, according to 'chrom' type) • starting position on the chromosome (`integer`, considered within) • ending position on the chromosome (`integer`, considered within) • chromosome index (`integer` vector, see below)

Details

The C code relies heavily on the ordering to fastly retrieve the elements that overlap the queried window. Elements entirely comprised in the window are returned, as well as elements that only partially overlap it.

Value

`subtrack` returns a single `data.frame` merging all columns provided, with the subset of rows corresponding to elements in the queried window. This `data.frame` has no row name, and is a valid track (See 'Track definition' for further details).

`sizetrack` returns a single `integer` value corresponding to the count of rows in the queried window.

`istrack` returns a single `TRUE` value if the data collection provided is a valid track. Otherwise it returns a single `FALSE` value, with a "why" attribute containing a single character string explaining the (first) condition that is not fulfilled.

Track definition

A track is defined as a `data.frame` with a variable amount of data (in columns) about a variable amount of features (in rows).

3 columns are mandatory, with restricted names and types :

chrom The chromosomal location of the feature, as `integer` or `factor`.

start The starting position of the feature on the chromosome, as `integer`.

end The ending position of the feature on the chromosome, as `integer`.

The track is supposed to be ordered by chromosome, then by starting position. When chromosomes are stored as `factors`, they need to be numerically ordered by their internal codes (as the `order` function does), not alphabetically by their labels.

Chromosome index

In order to guarantee good performances, chromosomes are to be indexed. As the rows are supposed to be ordered by chromosome, then by starting position (see 'Track definition'), reminding starting or ending rows of each chromosome can save huge amounts of computation time in large tracks.

The following specifications must be fulfilled :

- It must be an `integer` vector, with the last row index of each chromosome in the track indexed.
- Values are to be ordered by chromosome, in the same way than the 'chrom' column.
- For `integer` 'chrom', values are extracted by position (chromosome '1' is the first value ...).
- For `factor` 'chrom', values are extracted by names (named with 'chrom' levels).
- Chromosomes without data in the track must be described, with `NA integer` values.

See the 'Example' section below for index computation.

Note

These three functions are proposed for generic usage on `data.frame`, `list` or vectors. The `track.table` class implements more suitable `slice`, `size` and `check` methods, and handles autonomously the indexing.

Author(s)

Sylvain Mareschal

Examples

```
# Exemplar data : subset of human genes
data(hsFeatures)

# Track validity
print(istrack(hsGenes))
hsGenes <- hsGenes[ order(hsGenes$chrom, hsGenes$start) ,]
print(istrack(hsGenes))

# Chromosome index (factorial 'chrom')
index <- tapply(1:nrow(hsGenes), hsGenes$chrom, max)

# Factor chrom query
print(class(hsGenes$chrom))
subtrack("1", 10e6, 15e6, index, hsGenes)

# Row count
a <- nrow(subtrack("1", 10e6, 15e6, index, hsGenes))
b <- sizetrack("1", 10e6, 15e6, index, hsGenes)
if(a != b) stop("Inconsistency")

# Multiple sources
length <- hsGenes$end - hsGenes$start
subtrack("1", 10e6, 15e6, index, hsGenes, length)
subtrack("1", 10e6, 15e6, index, hsGenes, length=length)

# Speed comparison (x200 here)
system.time(
```

```
  for(i in 1:40000) {
    subtrack("1", 10e6, 15e6, index, hsGenes)
  }
)
system.time(
  for(i in 1:200) {
    hsGenes[ hsGenes$chrom == "1" & hsGenes$start <= 15e6 & hsGenes$end >= 10e6 ,]
  }
)

# Convert chrom from factor to integer
hsGenes$chrom <- as.integer(as.character(hsGenes$chrom))

# Chromosome index (integer 'chrom')
index <- rep(NA_integer_, 24)
tmpIndex <- tapply(1:nrow(hsGenes), hsGenes$chrom, max)
index[ as.integer(names(tmpIndex)) ] <- tmpIndex

# Integer chrom query
print(class(hsGenes$chrom))
subtrack(1, 10e6, 15e6, index, hsGenes)
```

`tk.browse` *R Genome browser*

Description

The `browsePlot` function produces an usual R plot from a `drawable` inheriting object list, at specific coordinates in the genome.

The `tk.browse` function summons a TCL-TK interface to navigate through the whole genome, relying on `browsePlot` for the plotting.

The former may be called directly to automatically export views from the genome browser, the latter is more suited to an interactive browsing with frequent coordinate jumps.

Usage

```
tk.browse(drawables, hscale = NA, vscale = NA, blocking = FALSE,
  scaleFactor = NA, updateLimit = 0.4)
browsePlot(drawables, chrom = NA, start = NA, end = NA,
  customLayout = FALSE, xaxt = "s", xaxm = 1.5, ...)
```

Arguments

`drawables`	A `drawable.list` object containing the tracks to plot, as returned by `drawable.list` or `tk.tracks`.
`hscale`	Single numeric value, to be passed to `tkrplot`.
`vscale`	Single numeric value, to be passed to `tkrplot`.
`blocking`	Single logical value, whether to wait for the interface window to be closed before unfreezing the R console. The `FALSE` default let you use R and the interface in parallel (see the 'Typing R commands while browsing' section below), the codeTRUE value is used essentially in the stand alone version.

scaleFactor	Single numeric value, scaling factor to convert pixel sizes in tkrplot scale units. NA will use the OS dependant default values.
updateLimit	Single numeric value, minimal time (in seconds) between two image updates when move or zoom key are continuously pressed. This is used to limit zoom and move speed on fast computers.
chrom	Single character value, the chromosome to plot.
start	Single integer value, the left boundary of the window to plot. NA will use 0.
end	Single integer value, the right boundary of the window to plot. NA will use the maximal drawable$getChromEnd() value throughout trackList.
customLayout	Single logical value, whether to organize the various plot or not. If FALSE, layout will be called and you will not be able to add custom plots. If TRUE, it is up to the user to define its layout call to handle the track plots that will be issued by the function.
xaxt	X axis showing (see par). Only applied to the last track, other ones never show their X axis ("n").
xaxm	Minimal bottom margin for the last track (see par). It assures enough margin is reserved for X axis showing (see xaxt).
...	Further arguments are passed through to the draw method of each track in trackList.

Value

tk.browse invisibly returns the drawables argument (See the 'Typing R commands while browsing' section above).

browsePlot invisibly returns the par function output for the last track plot, as it is used by tk.browse.

Interactive browsing

Jump to specific location: The left upper panel can be used to jump to specific coordinates, defined by a chromosome name, a starting position and an ending position (as floating point numerics in millions of base pairs.

Move along a chromosome: The left and right arrow keys may be used to shift the window to the corresponding side.

Generic zoom: The up and down arrow keys, as well as the vertical mouse wheel, may be used to zoom in or out on the current location.

Localized zoom in: A zoom can also be achieved with a mouse drag on the region to investigate : maintain a left click on the position to use as the new left boundary, and release the click at the position of the new right boundary.

Resize plot area: The plot area size is defined by hscale and vscale. During interactive browsing, resize the browser window and use the "r" key to adjust the plot area to the window size.

Typing R commands while browsing

`tk.browse` returns the `drawable.list` objects it uses to store currently browsed data. As it is a reference class object, the same memory location is shared by `tk.browse` and the returned object, so updates (like track addition or edition) made by `tk.browse` will impact the object in the R Command Line Interface (CLI), and updates made via R commands will impact the current `tk.browse` session.

Some sub-interfaces (like information pop-ups and track selection panels) may freeze the R command prompt while opened, make sure to have only the `tk.browse` main window opened when typing R commands.

Notice some operating systems (including Windows) restrain users to type R commands while a tcl-tk window is opened, or seems to be instable while doing so. Setting `blocking` to `TRUE` will enforce this behavior, keeping users from typing commands while `tk.browse` is running.

Author(s)

Sylvain Mareschal

See Also

`drawable`, `track.table`

`tk.convert` *RDT file conversion*

Description

This function provides a tcl-tk interface to convert RDT table files into CSV-like files, and to produce basic `track.table` objects from such files.

Usage

```
tk.convert(blocking = FALSE)
```

Arguments

`blocking`	Single logical value, whether to wait for the interface window to be closed before unfreezing the R console. The `FALSE` default let you use R and the interface in parallel, the codeTRUE is used essentially in the stand alone version.

Author(s)

Sylvain Mareschal

See Also

`track.table-class`

tk.tracks *Track management for Genome browser*

Description

This function allows to load and edit the drawable.list object that is to be passed to tk.browse and codebrowsePlot, using a Tcl-tk interface

Usage

```
tk.tracks(drawables = drawable.list(), parent = NULL)
```

Arguments

drawables	A previously built drawable.list to edit. Default value builds an empty one.
parent	An optional tktoplevel to bind message boxes to. Impact on the Tcl-tk interface behaviour is slight, so only advanced users should take care of it.

Value

Returns a drawable.list object. Notice that if 'drawables' was provided, it will also be updated "in-place" (standard reference class behavior).

Author(s)

Sylvain Mareschal

See Also

tk.browse, browsePlot, drawable.list-class, drawable-class, findDrawables

track-constructors *Track constructors*

Description

Produces track.table-inheriting objects.

Usage

```
track.table(..., .name, .parameters, .organism, .assembly, .chromosomes,
  .makeNames = FALSE, .orderCols = TRUE, warn = TRUE)
track.bam(bamPath, baiPath, addChr, quiet = FALSE, .name, .organism,
  .assembly, .parameters, warn = TRUE)
track.genes(...)
track.bands(...)
track.exons(...)
track.CNV(...)
```

Arguments

...	Arguments to be passed to the inherited constructor (refTable for track.table, track.table for the others). track.table has few more restrictions, see track.table-class for further details.
.name	Single character value, to fill the name field inherited from drawable.
warn	Single logical value, to be passed to the appropriate check method. Other conversion warnings rely on this value too in track.table.
.parameters	A list of drawing parameters, to fill the parameters field of the object.
.organism	Single character value, to fill the organism field of the object.
.assembly	Single character value, to fill the assembly field of the object.
.chromosomes	Single character value, levels to use for the 'chrom' column if conversion to factor is needed.
.makeNames	Single logical value, whether to compute the 'name' column with unique values or not. If TRUE, any existing 'name' column will be replaced.
.orderCols	Single logical value, whether to reorder the columns for more consistency between tracks or not.
bamPath	Single character value, the file name and path to a BAM file (.bam) to build a track around.
baiPath	Single character value, the file name and path to the corresponding BAI file (.bai) to build a track around. If missing, a guess will be tried (adding '.bai' to bamPath or remplacing '.bam' by '.bai').
addChr	Single logical value, whether to automatically add 'chr' ahead chromosome names when querying or not. If missing, a guess will be tried looking for chromosome names beginning by 'chr' in the BAM header declaration.
quiet	Single logical value, whether to throw diagnostic messages during BAI parsing or not.

Value

track.table and track.bam inheriting objects.

Author(s)

Sylvain Mareschal

See Also

track.table-class, track.bam-class

Examples

```
# track.table from a data.frame
df <- data.frame(chrom=1, strand="+", start=1:5, end=2:6, name=letters[1:5], stringsAsFactors=FALSE)
track.table(df)

# track.table from vectors
track.table(chrom=1, strand="+", start=1:5, end=2:6, name=letters[1:5])

# track.bam
track.bam(system.file("extdata/ATM.bam", package="Rgb"))
```

`track.bam-class` *Class* `"track.bam"`

Description

`"track.bam"` is a drawing wraper for Binary Alignment Map files (SAMtools).

Notice the data are not stored directly in the object, but stay in the original BAM file, thus exported `track.bam` objects may be broken (the `check` method can confirm this).

Objects are produced by the `track.bam` constructor.

Extends

Class `sliceable`, directly.
Class `drawable`, by class `sliceable`, distance 2.

All reference classes extend and inherit methods from `envRefClass`.

Fields

`addChr`**:** Single `logical` value, whether to automatically add 'chr' ahead chromosome names when querying or not..

`assembly`**:** Single `character` value, the assembly version for the coordinates stored in the object. Must have length 1, should not be `NA`.

`baiPath`**:** Single `character` value, the full path to the BAI index file in use.

`bamPath`**:** Single `character` value, the full path to the BAM file in use.

`header`**:** A `data.frame` describing the @SQ elements of the BAM header (one per row).

`index`**:** The parsed content of the BAI index, as a unamed `list` with one element by reference sequence, itself a `list` with 'bins' and 'intervals' elements. 'bins' is a named `list` of two-column matrices ('start' and 'end'), giving virtual BGZF coordinates of the described bin (as `double`). 'intervals' is a `double` vector of virtual BGZF coordinates, used for linear filtering (see SAM specification for further details).

`organism`**:** Single `character` value, the name of the organism whose data is stored in the object. Must have length 1, should not be `NA`.

The following fields are inherited (from the corresponding class):

- name (drawable)
- parameters (drawable)

Methods

`crawl(chrom, start, end, addChr = , verbosity = , ..., init, loop, final)`**:** Apply a custom processing to reads in a genomic window.
- **chrom** : single integer, numeric or character value, the chromosomal location. NA is not handled.
- **start** : single integer or numeric value, inferior boundary of the window. NA is not handled.
- **end** : single integer or numeric value, superior boundary of the window. NA is not handled.
- **addChr** : single logical value, whether to systematically add 'chr' in front of the 'chrom' value or not.
- **verbosity** : single integer value, the level of verbosity during processing (0, 1 or 2).

- **...** : arguments to be passed to 'init', 'loop' or 'final'.
- **init** : an expression to be evaluated before looping on reads.
- **loop** : a function with taking at least 'read' and '...' as arguments, modifying 'output' in the parent environment.
- **final** : an expression to be evaluated after looping on reads.

`depth(..., qBase = , qMap = ):` Counts covering bases for each genomic position, similarly to SAMtools' depth.
- **...** : argumets to be passed to the crawl() method.
- **qBase** : single integer value, minimal base quality for a base to be counted.
- **qMap** : single integer value, minimal mapping quality for a base to be counted.

`extract(...):` Extract reads as a list, similarly to SAMtools' view.
- **...** : argumets to be passed to the crawl() method.

`pileup(..., qBase = , qMap = ):` Counts each nucleotide type for each genomic position, similarly to SAMtools' mpileup.
- **...** : argumets to be passed to the crawl() method.
- **qBase** : single integer value, minimal base quality for a base to be counted.
- **qMap** : single integer value, minimal mapping quality for a base to be counted.

The following methods are inherited (from the corresponding class):

- callParams (drawable)
- callSuper (envRefClass)
- check (drawable)
- chromosomes (drawable)
- copy (envRefClass)
- defaultParams (sliceable, overloaded)
- draw (sliceable)
- export (envRefClass)
- field (envRefClass)
- fix.param (drawable)
- getChromEnd (sliceable, overloaded)
- getClass (envRefClass)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- import (envRefClass)
- initFields (envRefClass)
- initialize (drawable, overloaded)
- setName (drawable)
- setParam (drawable)
- show (sliceable, overloaded)
- slice (sliceable, overloaded)
- trace (envRefClass)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

References

http://samtools.github.io/hts-specs/SAMv1.pdf

See Also

track.table, sliceable-class, drawable-class

---

track.bands-class *Class* "track.bands"

Description

This class is a variation of the track.table class dedicated to cytogenetic banding, enforcing new drawing parameter defaults and a few specialized methods.

Objects can be created by two distincts means :

- Using the corresponding constructors, which work like the track.table constructor.
- Importing a track.table object in an empty object created by a call to new.

Extends

Class track.table, directly.
Class refTable, by class track.table, distance 2.
Class crossable, by class track.table, distance 2.
Class sliceable, by class track.table, distance 3.
Class drawable, by class track.table, distance 4.

All reference classes extend and inherit methods from envRefClass.

Fields

The following fields are inherited (from the corresponding class):

- assembly (track.table)
- checktrack (track.table)
- colCount (refTable)
- colIterator (refTable)
- colNames (refTable)
- colReferences (refTable)
- index (track.table)
- name (drawable)
- organism (track.table)
- parameters (drawable)
- rowCount (refTable)
- rowNamed (refTable)
- rowNames (refTable)
- subtrack (track.table)
- values (refTable)

Methods

The following methods are inherited (from the corresponding class):

- addArms (track.table)
- addColumn (track.table)
- addDataFrame (refTable)
- addEmptyRows (refTable)
- addList (track.table)
- addVectors (refTable)
- buildCalls (track.table)
- buildIndex (track.table)
- callParams (drawable)
- callSuper (envRefClass)
- check (track.table)
- chromosomes (track.table)
- coerce (track.table)
- colOrder (refTable)
- copy (refTable)
- cross (crossable)
- defaultParams (track.table, overloaded)
- delColumns (track.table)
- draw (sliceable)
- erase (refTable)
- eraseArms (track.table)
- export (envRefClass)
- extract (refTable)
- field (envRefClass)
- fill (track.table)
- fix.param (drawable)
- getChromEnd (track.table)
- getClass (envRefClass)
- getColCount (refTable)
- getColNames (refTable)
- getLevels (refTable)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- getRowCount (refTable)
- getRowNames (refTable)
- import (envRefClass)
- indexes (refTable)

- initFields (envRefClass)
- initialize (track.table)
- isArmed (track.table)
- metaFields (track.table)
- rowOrder (track.table)
- setColNames (track.table)
- setLevels (track.table)
- setName (drawable)
- setParam (drawable)
- setRowNames (refTable)
- show (track.table, overloaded)
- slice (track.table)
- trace (envRefClass)
- types (refTable)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

track.table-class, track.table

track.bands, track.bands.UCSC

---

track.CNV-class *Class "track.CNV"*

Description

This class is a variation of the track.table class dedicated to constitutive Copy Number Variations, enforcing new drawing parameter defaults and a few specialized methods.

Objects can be created by two distincts means :

- Using the corresponding constructors, which work like the track.table constructor.
- Importing a track.table object in an empty object created by a call to new.

Extends

Class track.table, directly.
Class refTable, by class track.table, distance 2.
Class crossable, by class track.table, distance 2.
Class sliceable, by class track.table, distance 3.
Class drawable, by class track.table, distance 4.

All reference classes extend and inherit methods from envRefClass.

Fields

The following fields are inherited (from the corresponding class):

- assembly (track.table)
- checktrack (track.table)
- colCount (refTable)
- colIterator (refTable)
- colNames (refTable)
- colReferences (refTable)
- index (track.table)
- name (drawable)
- organism (track.table)
- parameters (drawable)
- rowCount (refTable)
- rowNamed (refTable)
- rowNames (refTable)
- subtrack (track.table)
- values (refTable)

Methods

The following methods are inherited (from the corresponding class):

- addArms (track.table)
- addColumn (track.table)
- addDataFrame (refTable)
- addEmptyRows (refTable)
- addList (track.table)
- addVectors (refTable)
- buildCalls (track.table)
- buildIndex (track.table)
- callParams (drawable)
- callSuper (envRefClass)
- check (track.table)
- chromosomes (track.table)
- coerce (track.table)
- colOrder (refTable)
- copy (refTable)
- cross (crossable)
- defaultParams (track.table, overloaded)
- delColumns (track.table)
- draw (sliceable)
- erase (refTable)

- eraseArms (track.table)
- export (envRefClass)
- extract (refTable)
- field (envRefClass)
- fill (track.table)
- fix.param (drawable)
- getChromEnd (track.table)
- getClass (envRefClass)
- getColCount (refTable)
- getColNames (refTable)
- getLevels (refTable)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- getRowCount (refTable)
- getRowNames (refTable)
- import (envRefClass)
- indexes (refTable)
- initFields (envRefClass)
- initialize (track.table)
- isArmed (track.table)
- metaFields (track.table)
- rowOrder (track.table)
- setColNames (track.table)
- setLevels (track.table)
- setName (drawable)
- setParam (drawable)
- setRowNames (refTable)
- show (track.table, overloaded)
- slice (track.table)
- trace (envRefClass)
- types (refTable)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

`track.table-class`, `track.table`, `track.table.GTF`

`track.CNV`, `track.CNV.DGV`

`track.exons-class` *Class* `"track.exons"`

Description

This class is a variation of the `track.table` class dedicated to exons, enforcing new drawing parameter defaults and a few specialized methods.

Objects can be created by two distincts means :

- Using the corresponding constructors, which work like the `track.table` constructor.
- Importing a `track.table` object in an empty object created by a call to new (see Examples).

Extends

Class `track.table`, directly.
Class `refTable`, by class `track.table`, distance 2.
Class `crossable`, by class `track.table`, distance 2.
Class `sliceable`, by class `track.table`, distance 3.
Class `drawable`, by class `track.table`, distance 4.

All reference classes extend and inherit methods from `envRefClass`.

Fields

The following fields are inherited (from the corresponding class):

- assembly (track.table)
- checktrack (track.table)
- colCount (refTable)
- colIterator (refTable)
- colNames (refTable)
- colReferences (refTable)
- index (track.table)
- name (drawable)
- organism (track.table)
- parameters (drawable)
- rowCount (refTable)
- rowNamed (refTable)
- rowNames (refTable)
- subtrack (track.table)
- values (refTable)

Methods

The following methods are inherited (from the corresponding class):

- addArms (track.table)
- addColumn (track.table)
- addDataFrame (refTable)
- addEmptyRows (refTable)
- addList (track.table)
- addVectors (refTable)
- buildCalls (track.table)
- buildIndex (track.table)
- callParams (drawable)
- callSuper (envRefClass)
- check (track.table)
- chromosomes (track.table)
- coerce (track.table)
- colOrder (refTable)
- copy (refTable)
- cross (crossable)
- defaultParams (track.table, overloaded)
- delColumns (track.table)
- draw (sliceable)
- erase (refTable)
- eraseArms (track.table)
- export (envRefClass)
- extract (refTable)
- field (envRefClass)
- fill (track.table)
- fix.param (drawable)
- getChromEnd (track.table)
- getClass (envRefClass)
- getColCount (refTable)
- getColNames (refTable)
- getLevels (refTable)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- getRowCount (refTable)
- getRowNames (refTable)
- import (envRefClass)
- indexes (refTable)

- initFields (envRefClass)
- initialize (track.table)
- isArmed (track.table)
- metaFields (track.table)
- rowOrder (track.table)
- setColNames (track.table)
- setLevels (track.table)
- setName (drawable)
- setParam (drawable)
- setRowNames (refTable)
- show (track.table, overloaded)
- slice (track.table)
- trace (envRefClass)
- types (refTable)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

`track.table-class`, `track.table`, `track.table.GTF`

`track.exons`, `track.exons.CCDS`

---

track.fasta-class *Class* "track.fasta"

Description

"track.fasta" is a drawing wraper for FASTA files.

Notice the data are not stored directly in the object, but stay in the original FASTA file(s), thus exported track.fasta objects may be broken (the check method can confirm this).

Objects are produced by the `track.fasta.multi` and `track.fasta.collection` constructors.

Extends

Class `sliceable`, directly.
Class `drawable`, by class `sliceable`, distance 2.

All reference classes extend and inherit methods from `envRefClass`.

Fields

`assembly`: Single `character` value, the assembly version for the coordinates stored in the object. Must have length 1, should not be `NA`.

`files`: A `data.frame`, with 6 columns : file (character), header (character), startOffset (numeric), lineLength (integer), breakSize (integer) and contentSize (integer). Each row refers to a distinct chromosome, whose name is stored as row name.

`organism`: Single `character` value, the name of the organism whose data is stored in the object. Must have length 1, should not be `NA`.

The following fields are inherited (from the corresponding class):

- name (drawable)
- parameters (drawable)

Methods

The following methods are inherited (from the corresponding class):

- callParams (drawable)
- callSuper (envRefClass)
- check (drawable, overloaded)
- chromosomes (drawable, overloaded)
- copy (envRefClass)
- defaultParams (sliceable, overloaded)
- draw (sliceable)
- export (envRefClass)
- field (envRefClass)
- fix.param (drawable)
- getChromEnd (sliceable, overloaded)
- getClass (envRefClass)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- import (envRefClass)
- initFields (envRefClass)
- initialize (drawable, overloaded)
- setName (drawable)
- setParam (drawable)
- show (sliceable, overloaded)
- slice (sliceable, overloaded)
- trace (envRefClass)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

track.table, sliceable-class, drawable-class

---

track.fasta-constructors

Track constructors for FASTA files

Description

Produces sliceable-inheriting objects to query "in situ" FASTA files.

track.fasta.multi is designed to handle a single multi-FASTA file aggregating all the chromosomes of an organism. An index as generated by the HTSlib (formerly "SAMtools") faidx command is required.

track.fasta.collection is designed to handle a collection of standard FASTA files, one per chromosome.

Usage

```
track.fasta.multi(fastaFile, indexFile, .name, .organism, .assembly,
  .parameters, warn = TRUE)
track.fasta.collection(files, chromosomes, .name, .organism, .assembly,
  .parameters, warn = TRUE)
```

Arguments

fastaFile	Single character value, the file name and path to a multi-FASTA file (.fa) to build a track upon.
indexFile	Single character value, the file name and path to a multi-FASTA index file (.fai) corresponding to fastaFile. This index file is supposd to be produced by the faidx command from SAMtools.
.name	Single character value, to fill the name field inherited from drawable.
.organism	Single character value, to fill the organism field of the object.
.assembly	Single character value, to fill the assembly field of the object.
.parameters	A list of drawing parameters, to fill the parameters field of the object.
warn	Single logical value, to be passed to the appropriate check method. Other conversion warnings rely on this value too in track.table.
files	Character vector, file names and paths to multiple single-FASTA file (.fa) to build a track upon (one per chromosome).
chromosomes	Character vector, chromosomes names to attribute to each file in files.

Details

Both functions suppose the FASTA files to respect the following :

- They begin with a single line of comment, after a '>' sign.
- All sequence lines have the same length (whatever it is).
- The line separator (\n, \r\n) is always the same in a file.
- No empty line.

Returned sequences may be wrong (without error or notice !) if any of these is not fullfilled. Standard sources (see below) usually enforce these conditions.

Value

An object of class `track.fasta`.

Author(s)

Sylvain Mareschal

References

Example of FASTA file collection (human assembly 'hg19'), from UCSC : http://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz

Example of single multi-FASTA file (human assembly 'hg19'), from 1000 genomes : ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/human_g1k_v37.fasta.gz

The faidx documentation, from the HTSlib project.

See Also

`track-constructors`, `Annotation`

`tk.browse`, `browsePlot`

---

track.genes-class *Class* "track.genes"

Description

This class is a variation of the `track.table` class dedicated to genes, enforcing new drawing parameter defaults and a few specialized methods.

Objects can be created by two distincts means :

- Using the corresponding constructors, which work like the `track.table` constructor.
- Importing a `track.table` object in an empty object created by a call to new (see Examples).

Extends

Class `track.table`, directly.
Class `refTable`, by class `track.table`, distance 2.
Class `crossable`, by class `track.table`, distance 2.
Class `sliceable`, by class `track.table`, distance 3.
Class `drawable`, by class `track.table`, distance 4.

All reference classes extend and inherit methods from `envRefClass`.

Fields

The following fields are inherited (from the corresponding class):

- assembly (track.table)
- checktrack (track.table)
- colCount (refTable)
- colIterator (refTable)
- colNames (refTable)
- colReferences (refTable)
- index (track.table)
- name (drawable)
- organism (track.table)
- parameters (drawable)
- rowCount (refTable)
- rowNamed (refTable)
- rowNames (refTable)
- subtrack (track.table)
- values (refTable)

Methods

The following methods are inherited (from the corresponding class):

- addArms (track.table)
- addColumn (track.table)
- addDataFrame (refTable)
- addEmptyRows (refTable)
- addList (track.table)
- addVectors (refTable)
- buildCalls (track.table)
- buildIndex (track.table)
- callParams (drawable)
- callSuper (envRefClass)
- check (track.table)
- chromosomes (track.table)
- coerce (track.table)
- colOrder (refTable)
- copy (refTable)
- cross (crossable)
- defaultParams (track.table)
- delColumns (track.table)
- draw (sliceable)
- erase (refTable)

- eraseArms (track.table)
- export (envRefClass)
- extract (refTable)
- field (envRefClass)
- fill (track.table)
- fix.param (drawable)
- getChromEnd (track.table)
- getClass (envRefClass)
- getColCount (refTable)
- getColNames (refTable)
- getLevels (refTable)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- getRowCount (refTable)
- getRowNames (refTable)
- import (envRefClass)
- indexes (refTable)
- initFields (envRefClass)
- initialize (track.table)
- isArmed (track.table)
- metaFields (track.table)
- rowOrder (track.table)
- setColNames (track.table)
- setLevels (track.table)
- setName (drawable)
- setParam (drawable)
- setRowNames (refTable)
- show (track.table, overloaded)
- slice (track.table)
- trace (envRefClass)
- types (refTable)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

`track.table-class`, `track.table`, `track.table.GTF`

`track.genes`, `track.genes.NCBI`

track.table-class *Class* "track.table"

Description

"track.table" describes a collection of features localized on a genome, defined by a chromosomal location ("chrom"), two boundaries ("start" and "end") and various data (other columns).

Objects can be created by two distincts means :

- The track.table constructor, similar to the data.frame constructor. It imports a single data.frame or a collection of vectors into the object, reorder it to meet the track.table class restrictions (see below) and check immediatly for validity.
- The new function (standard behavior for S4 and reference classes), which produces an empty object and do NOT check for validity. It takes as named arguments the values to store in the fields of the various classes the new object will inherit from.

3 columns are mandatory in "track.table", with restricted names and types :

name The unique name of the feature, as character.

chrom The chromosomal location of the feature, as integer or factor.

start The starting position of the feature on the chromosome, as integer.

end The ending position of the feature on the chromosome, as integer.

The track is supposed to be ordered by chromosome, then by starting position. When chromosomes are stored as factors, they need to be numerically ordered by their internal codes (as the order function does), not alphabetically by their labels.

As the slice method relies on a row index, each update in the feature coordinates must be followed by a call to the buildIndex method, a behavior that is enforced by overloading most of refTable-inherited methods.

As the slice method relies on a R call object using refTable column references, each update in the column names must be followed by a call to the buildCalls method, a behavior that is enforced by overloading most of refTable-inherited methods.

Extends

Class refTable, directly.
Class crossable, directly.
Class sliceable, by class crossable, distance 2.
Class drawable, by class crossable, distance 3.

All reference classes extend and inherit methods from envRefClass.

Fields

assembly: Single character value, the assembly version for the coordinates stored in the object. Must have length 1, should not be NA.

checktrack: A call to the C external "checktrack", for faster object check (see the check method).

index: integer vector, giving the index of the first row in each chromosome. See the subtrack function for further details.

`organism`: Single `character` value, the name of the organism whose data is stored in the object. Must have length 1, should not be `NA`.

`subtrack`: A `call` to the C external "subtrack", for faster slicing (see the `slice` method).

The following fields are inherited (from the corresponding class):

- colCount (refTable)
- colIterator (refTable)
- colNames (refTable)
- colReferences (refTable)
- name (drawable)
- parameters (drawable)
- rowCount (refTable)
- rowNamed (refTable)
- rowNames (refTable)
- values (refTable)

Methods

`addArms(centromeres, temp = )`: Adds an arm localization ('p' or 'q') to the 'chrom' column.
- **centromeres** : named numeric vector, providing the centromere position of each chromosome. Can also be a band track, as returned by track.UCSC_bands().
- **temp** : single logical value, whether to alter the object or return an altered copy.

`buildCalls()`: Updates 'checktrack' and 'subtrack' fields. To be performed after each modification of colNames and colReferences (concerned methods are overloaded to enforce this).

`buildGroupPosition(groupBy, colName = , reverse = )`: Adds a column to be used as 'groupPosition' by draw.boxes()
- **groupBy** : single character value, the name of a column to group rows on.
- **colName** : single character value, the name of the column to buid.
- **reverse** : single logical value, whether to reverse numbering on reverse strand or not.

`buildGroupSize(groupBy, colName = )`: Adds a column to be used as 'groupSize' by draw.boxes()
- **groupBy** : single character value, the name of a column to group rows on.
- **colName** : single character value, the name of the column to buid.

`buildIndex()`: Updates the 'index' parameter, should be done after any change made on the 'chrom' column (concerned methods are overloaded to enforce this).

`eraseArms(temp = )`: Removes 'p' and 'q' added by the addArms() method from the 'chrom' column.
- **temp** : single logical value, whether to alter the object or return an altered copy.

`isArmed()`: Detects whether the 'chrom' column refers to whole chromosomes or chromosome arms.

`segMerge(...)`: Apply the segMerge() function to the track content. - **...** : arguments to be passed to segMerge().

The following methods are inherited (from the corresponding class):

- addColumn (refTable, overloaded)
- addDataFrame (refTable)
- addEmptyRows (refTable)
- addList (refTable, overloaded)

- addVectors (refTable)
- callParams (drawable)
- callSuper (envRefClass)
- check (refTable, overloaded)
- chromosomes (drawable, overloaded)
- coerce (refTable, overloaded)
- colOrder (refTable)
- copy (refTable)
- cross (crossable)
- defaultParams (sliceable, overloaded)
- delColumns (refTable, overloaded)
- draw (sliceable)
- erase (refTable)
- export (envRefClass)
- extract (refTable)
- field (envRefClass)
- fill (refTable, overloaded)
- fix.param (drawable)
- getChromEnd (sliceable, overloaded)
- getClass (envRefClass)
- getColCount (refTable)
- getColNames (refTable)
- getLevels (refTable)
- getName (drawable)
- getParam (drawable)
- getRefClass (envRefClass)
- getRowCount (refTable)
- getRowNames (refTable)
- import (envRefClass)
- indexes (refTable)
- initFields (envRefClass)
- initialize (refTable, overloaded)
- metaFields (refTable, overloaded)
- rowOrder (refTable, overloaded)
- setColNames (refTable, overloaded)
- setLevels (refTable, overloaded)
- setName (drawable)
- setParam (drawable)
- setRowNames (refTable)
- show (refTable, overloaded)
- slice (sliceable, overloaded)
- trace (envRefClass)
- types (refTable)
- untrace (envRefClass)
- usingMethods (envRefClass)

Author(s)

Sylvain Mareschal

See Also

refTable-class

track.table, subtrack, istrack

Examples

```
# Exemplar data : subset of human genes
data(hsFeatures)

# Construction
trackTable <- track.table(
  hsGenes,
  .name = "NCBI Genes",
  .organism = "Homo sapiens",
  .assembly = "GRCh37"
)

# Slicing
print(trackTable$slice(chrom="1", as.integer(15e6), as.integer(20e6)))
```

Index