Package 'gtrellis'

April 23, 2016

Type Package

Title Genome Level Trellis Layout

Version 1.2.0

Date 2015-8-22

Author Zuguang Gu

Maintainer Zuguang Gu <z.gu@dkfz.de>

Depends R (>= 3.1.2), grid

Imports circlize (>= 0.2.3), GetoptLong

Suggests testthat (>= 0.3), knitr, GenomicRanges, RColorBrewer, markdown

VignetteBuilder knitr

Description Genome level Trellis graph visualizes genomic data conditioned by genomic categories (e.g. chromosomes). For each genomic category, multiple dimensional data which are represented as tracks describe different features from different aspects. This package provides high flexibility to arrange genomic categories and add self-defined graphics in the plot.

biocViews Software, Visualization, Sequencing

URL https://github.com/jokergoo/gtrellis

License GPL (>= 2)

Repository Bioconductor

Date/Publication 2015-8-22 00:00:00

NeedsCompilation no

R topics documented:

add_ideogram_track *Add ideogram track*

Description

Add ideogram track

Usage

```
add_ideogram_track(cytoband = paste0(system.file(package = "circlize"),
   "/extdata/cytoBand.txt"), species = NULL, track = get_cell_meta_data("track") + 1)
```

Arguments

cytoband	Path of the cytoband file or a data frame that already contains cytoband data. Pass to read.cytoband.
species	Abbreviations of species. e.g. hg19 for human, mm10 for mouse. If this value is specified, the function will download cytoBand.txt.gz from UCSC ftp automatically. Pass to read.cytoband.
track	which track the ideogram is added in. By default it is the next track in the layout.

Details

A track which contains ideograms will be added to the plot.

The function tries to download cytoband file from UCSC ftp. If there is no cytoband file available for the species, there will be an error.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
gtrellis_layout(n_track = 2, ncol = 3,
    track_height = unit.c(unit(1, "null"), unit(5, "mm")))
add_ideogram_track(track = 2)
```

add_track *Add self-defined graphics track by track*

Description

Add self-defined graphics track by track

Usage

```
add_track(gr = NULL, category = NULL, track = get_cell_meta_data("track") + 1,
    clip = TRUE, panel.fun = function(gr) NULL)
```

Arguments

gr	genomic regions. It should be a data frame in BED format or a GRanges object.
category	subset of categories (e.g. chromosomes) that users want to add graphics. The value can be a vector which contains more than one category. By default it is all available categories.
track	which track the graphics will be added to. By default it is the next track. The value should only be a scalar.
clip	whether graphics are restricted inside the cell.
panel.fun	self-defined panel function to add graphics in each 'cell'. THe argument gr in panel.fun only contains data for the current category which is a subset of the main gr.

Details

Initialization of the Trellis layout and adding graphics are two independent steps. Once the layout initialization finished, each cell will be an independent plotting region. As same as panel.fun in circlize-package, the self-defined function panel.fun will be applied on every cell in the specified track (by default it is the 'current' track).

When adding graphics in each cell, get_cell_meta_data can return several meta data for the current cell.

Since this package is implemented by the grid graphic system, grid-family functions (such as grid.points, grid.rect, ...) should be used to add graphics. The usage of grid functions is quite similar as the traditional graphic functions. Followings are several examples:

```
grid.points(x, y)
grid.lines(x, y)
grid.rect(x, y, width, height)
```

Graphical parameters are usually passed by gpar:

```
    grid.points(x, y, gp = gpar(col = "red")
    grid.rect(x, y, width, height, gp = gpar(fill = "black", col = "red"))
```

grid system also support a large number of coordinate measurement systems by defining proper unit object which provides high flexibility to place graphics on the plotting regions.

```
    grid.points(x, y, default.units = "npc")
    grid.rect(x, y, width = unit(1, "cm"))
```

You can refer to the documentations and vignettes of grid-package to get a overview.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

get_cell_meta_data

Examples

```
require(circlize)
bed = circlize::generateRandomBed()
gtrellis_layout(track_ylim = range(bed[[4]]))
add_track(bed, panel.fun = function(bed) {
    x = (bed[[2]] + bed[[3]]) / 2
    y = bed[[4]]
    grid.points(x, y, pch = 16, size = unit(0.5, "mm"))
})

# you can add graphics in any cell by specifying `category` and `track`
all_chr = paste0("chr", 1:22)
letter = strsplit("MERRY CHRISTMAS!", "")[[1]]
gtrellis_layout(nrow = 5)
for(i in seq_along(letter)) {
    add_track(category = all_chr[i], track = 1, panel.fun = function(gr) {
        grid.text(letter[i], gp = gpar(fontsize = 30))
    })
}
```

`get_cell_meta_data` *Get meta data in a cell*

Description

Get meta data in a cell

Usage

```
get_cell_meta_data(name, category, track)
```

Arguments

`name`	name of the supported meta data, see 'details' section.
`category`	which category. By default it is the current category.
`track`	which track. By default it is the current track.

Details

Following meta data can be retrieved:

name name of the category.

xlim xlim without including padding. Cells in the same column share the same `xlim`.

ylim ylim without including padding.

extended_xlim xlim with padding.

extended_ylim ylim with padding.

original_xlim xlim in original data.

original_ylim ylim in original data.

column which column in the layout.

row which row in the layout.

track which track in the layout.

The vignette has a graphical explanation of all these meta data.

Value

Corresponding meta data that user queried.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
gtrellis_layout(ncol = 4, n_track = 3)
add_track(panel.fun = function(gr) {
    print(get_cell_meta_data("xlim"))
    print(get_cell_meta_data("ylim"))
    print(get_cell_meta_data("extended_xlim"))
    print(get_cell_meta_data("extended_ylim"))
    print(get_cell_meta_data("original_xlim"))
    print(get_cell_meta_data("original_ylim"))
    print(get_cell_meta_data("name"))
    print(get_cell_meta_data("column"))
    print(get_cell_meta_data("row"))
    print(get_cell_meta_data("track"))
    cat("\n\n")
})

for(chr in paste0("chr", 1:22)) {
    print(get_cell_meta_data("xlim", category = chr, track = 1))
    print(get_cell_meta_data("ylim", category = chr, track = 1))
    print(get_cell_meta_data("extended_xlim", category = chr, track = 1))
    print(get_cell_meta_data("extended_ylim", category = chr, track = 1))
    print(get_cell_meta_data("original_xlim", category = chr, track = 1))
    print(get_cell_meta_data("original_ylim", category = chr, track = 1))
    print(get_cell_meta_data("name", category = chr, track = 1))
    print(get_cell_meta_data("column", category = chr, track = 1))
    print(get_cell_meta_data("row", category = chr, track = 1))
    print(get_cell_meta_data("track", category = chr, track = 1))
    cat("\n\n")
}
```

---

gtrellis_layout *Initialize genome-level Trellis layout*

Description

Initialize genome-level Trellis layout

Usage

```
gtrellis_layout(data = NULL, category = NULL,
    species = NULL, nrow = NULL, ncol = NULL,
    n_track = 1, track_height = 1, track_ylim = c(0, 1),
    track_axis = TRUE, track_ylab = "",
    title = NULL, xlab = "Genomic positions", xaxis = TRUE,
    equal_width = FALSE, border = TRUE, asist_ticks = TRUE,
    xpadding = c(0, 0), ypadding = c(0, 0), gap = unit(1, "mm"),
    byrow = TRUE, newpage = TRUE, add_name_track = FALSE,
    name_fontsize = 10, name_track_fill = "#EEEEEE",
    add_ideogram_track = FALSE, ideogram_track_height = unit(2, "mm"),
    axis_label_fontsize = 6, lab_fontsize = 10, title_fontsize = 16)
```

Arguments

data	a data frame with at least three columns. The first three columns should be genomic categories (e.g. chromosomes), start positions and end positions. This data frame is used to extract ranges for each genomic category (minimal and maximal positions are taken as the range in the corresponding category).
category	subset of categories. It is also used for ordering.
species	Abbreviations of species. e.g. hg19 for human, mm10 for mouse. If this value is specified, the function will download `chromInfo.txt.gz` from UCSC ftp automatically. Short scaffolds will be removed if they have obvious different length as others. The argument is passed to `read.chromInfo`.
nrow	Number of rows in the layout.
ncol	Number of columns in the layout.
n_track	Number of tracks in each genomic category.
track_height	height of tracks. It should be numeric which means the value is relative and will be scaled into percent, or a `unit` object.
track_ylim	ranges on y axes of tracks. The value can be a vector of length two which means all tracks share same y ranges, or a matrix with two columns, or a vector of length `2*n_track` which will be coerced into the two-column matrix by rows.
track_axis	whether show y axes for tracks. The value is logical that can be either length one or number of tracks.
track_ylab	labels for tracks on y axes. The value can be either length one or number of tracks.
title	title of the plot.
xlab	labels on x axes.
xaxis	whether show x axes.
equal_width	whether all columns in the layout have the same width. If `TRUE`, short categories will be extended according to the longest category.
border	whether show borders.
asist_ticks	if axes ticks are added on one side in rows or columns, whether add ticks on the other sides.
xpadding	padding on x axes in each cell. Numeric value means relative ratio corresponding to the cell width. Use `I` to set it as absolute value which is measured in the data viewport (the coordinate system corresponding to the real data). Currently you cannot set it as a `unit` object.
ypadding	padding on y axes in each cell. Only numeric value is allowed currently.
gap	0 or a `unit` object. If it is length two, the first element corresponds to the gaps between rows and the second corresponds to the gaps between columns.
byrow	arrange categories (e.g. chromosomes) by rows or by columns in the layout.
newpage	whether call `grid.newpage` to create a new page.
add_name_track	whether add a pre-defined name track (insert before the first track). The name track is simply a track which only contains text. The default style of the name track is simple, but users can self define their own by `add_track`.

`name_fontsize` font size for text in the name track. Note the font size also affects the height of name track.

`name_track_fill`
filled color for name track.

`add_ideogram_track`
whether to add a pre-defined ideogram track (insert after the last track). If the cytoband data for specified species is not available, this argument is ignored. The ideogram track simply contains rectangles with different colors, implemented by `add_track`.

`ideogram_track_height`
Height of ideogram track. The value should be a `unit` object.

`axis_label_fontsize`
font size for axis labels.

`lab_fontsize` font size for x-labels and y-labels.

`title_fontsize` font size for title.

Details

Genome-level Trellis graph visualizes genomic data conditioned by genomic categories (e.g. chromosomes). For each genomic category, multiple dimensional data which are represented as tracks describe different features from different aspects. The `gtrellis_layout` function arranges genomic categories on the plot in a quite flexible way. Then users apply `add_track` to add self-defined graphics to the plot track by track.

For more detailed demonstration of the function, please go to the vignette.

Value

No value is returned.

Legend

Legend is not supported in this package. But it is easy to add legends based on the `grid` graphic system. Following example shows adding a simple legend on the right of the Trellis plot.

First create a `grob` object that contains the legend. The most simple way is to use `legendGrob` to construct a simple legend.

```
legd = legendGrob("label", pch = 16)
```

Create a layout which contains two columns and we set the width for the second column to the width of the legend by `grobWidth`.

```
layout = grid.layout(nrow = 1, ncol = 2, widths = unit.c(unit(1, "null"), grobWidth(legd)))
grid.newpage()
pushViewport(viewport(layout = layout))
```

In the left column, we add Trellis plot. Here you need to specify newpage to FALSE so that the plot is added into the current page which not creating a new one.

```
pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 1))
gtrellis_layout(nrow = 5, byrow = FALSE, track_ylim = range(bed[[4]]), newpage = FALSE)
add_track(bed, panel.fun = function(bed) {
    x = (bed[[2]] + bed[[3]]) / 2
    y = bed[[4]]
    grid.points(x, y, pch = 16, size = unit(0.5, "mm"))
})
upViewport()
```

In the right column, add the legend.

```
pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 2))
grid.draw(legd)
upViewport()
```

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

add_track, add_ideogram_track

Examples

```
gtrellis_layout()
gtrellis_layout(ncol = 5)
gtrellis_layout(n_track = 3, ncol = 4)

# for more examples, please go to the vignette
```

---

gtrellis_show_index *Show index on each cell*

Description

Show index on each cell

Usage

```
gtrellis_show_index()
```

Details

The function adds name and index of track for each cell. It is only for demonstration purpose.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
gtrellis_layout()
gtrellis_show_index()
```

Index