Package: CopyNumberPlots (via r-universe)

November 8, 2026

Type Package

Title Create Copy-Number Plots using karyoploteR functionality

Version 1.28.0

Author Bernat Gel <bgel@igtp.cat> and Miriam Magallon <mmagallon@igtp.cat>

Maintainer Bernat Gel <bgel@igtp.cat>

Description CopyNumberPlots have a set of functions extending karyoploteRs functionality to create beautiful, customizable and flexible plots of copy-number related data.

Depends R (>= 3.6), karyoploteR

Imports regioneR, IRanges, Rsamtools, SummarizedExperiment, VariantAnnotation, methods, stats, GenomeInfoDb, GenomicRanges, cn.mops, rhdf5, utils

Suggests BiocStyle, knitr, rmarkdown, panelcn.mops, BSgenome.Hsapiens.UCSC.hg19.masked, DNAcopy, testthat

License Artistic-2.0

URL https://github.com/bernatgel/CopyNumberPlots

BugReports https://github.com/bernatgel/CopyNumberPlots/issues

Encoding UTF-8

VignetteBuilder knitr

biocViews Visualization, CopyNumberVariation, Coverage, OneChannel, DataImport, Sequencing, DNASeq

LazyData false

RoxygenNote 7.1.1

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

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:50:05 UTC

RemoteUrl https://github.com/bioc/CopyNumberPlots

RemoteRef RELEASE_3_23

RemoteSha ccc507ce1e66a78a7dd7b69fba3fc3e189361803

Contents

colByCopyNumber . . . 3
computeHclustPlotOrder . . . 4
EnsemblStyle . . . 5
getBAFColumn . . . 5
getChrColumn . . . 6
getColumn . . . 7
getCopyNumberColors . . . 9
getCopyNumberColumn . . . 10
getEndColumn . . . 11
getIDColumn . . . 12
getLOHColumn . . . 13
getLRRColumn . . . 14
getPosColumn . . . 15
getSegmentValueColumn . . . 16
getStartColumn . . . 17
loadCopyNumberCalls . . . 18
loadCopyNumberCallsCnmops . . . 19
loadCopyNumberCallsCNVkit . . . 20
loadCopyNumberCallsDECoN . . . 21
loadCopyNumberCallsDNAcopy . . . 22
loadCopyNumberCallsPanelcnmops . . . 24
loadCopyNumberCallspennCNV . . . 25
loadCopyNumberCallsSeg . . . 26
loadSNPData . . . 27
loadSNPDataDNAcopy . . . 28
loadSNPDataFromVCF . . . 29
plotBAF . . . 30
plotCopyNumberCalls . . . 32
plotCopyNumberCallsAsLines . . . 34
plotCopyNumberSummary . . . 36
plotLRR . . . 38
plotSingleCellCopyNumberCalls . . . 40
plotSingleCellCopyNumberSummary . . . 42
prepareLabels . . . 43
readHDF5Ztree . . . 44
removeNAs . . . 45
transformChr . . . 46
UCSCStyle . . . 47
Ztree2Hclust . . . 47

Index **49**

colByCopyNumber *colByCopyNumber*

Description

Assign a color to each data element according to the copy number status of the its genomic location

Usage

```
colByCopyNumber(data, cn.calls, cn.column="cn", cn.colors=NULL)
```

Arguments

data	The data elements. Can be a GRanges or anything accepted by toGRanges
cn.calls	The CN calls to use to select the color for the data elements
cn.column	The name of the column with the copy number information. (defaults to "cn")
cn.colors	The specification of the colors. Internally, it used to call the function getCopyNumberColors. Detailed documentation on color specification can be found there. (defaults to NULL)

Details

Given a set of data elements positioned on the genome (either a GRanges or any other format accepted by regioneR's toGRanges) and a set of copy number calls, assign a color to each element depending on the copy number status of the genomic region where it's located. The colors can be customized using the cn.colors parameter as documented in getCopyNumberColors.

Value

Return a vector of colors with the same length as the number of elements in data

See Also

toGRanges, getCopyNumberColors, plotBAF

Examples

```
pos <- sort(floor(runif(1000, 1, 10000)))
baf.data <- toGRanges("chr1", pos, pos)
baf.data$baf <- rnorm(1000, mean = 0.5, sd = 0.05)
baf.data$baf[1:400] <- baf.data$baf[1:400] + c(0.2, -0.2)

breakpoint <- start(baf.data)[400]
cn.calls <- toGRanges(data.frame(chr=c("chr1", "chr1"), start=c(1, breakpoint+1), end=c(breakpoint, 10000), cn=c(

kp <- plotKaryotype(zoom=toGRanges("chr1", 1, 10000))
plotBAF(kp, baf.data, points.col = colByCopyNumber(baf.data, cn.calls))
```

computeHclustPlotOrder *computeHclustPlotOrder*

Description

Given the merge matrix of an hclust object, compute a valid ordering of the samples so a dendogram can be plotted with no line crossings.

Note: It's been implemented from scratch but in our experience reproduces exactly the ordering produced by R own ordering algorithm used by hclust.

Usage

```
computeHclustPlotOrder(m)
```

Arguments

m	(matrix) A merge matrix with 2 columns representing the merges bewteen elements and clusters. The format is described in the documentation of hclust.

Value

A vector with the ordering of the nodes.

Examples

```
ztree <- matrix(c(0,3,0.1,2,
                  1,4,0.2,2,
                  5,2,0.3,2,
                  6,7,0.4,2),
                  nrow=4)

 hc.tree <- Ztree2Hclust(ztree)
 plot(hc.tree)

 #Set the ordering to a non-correct ordering
 hc.tree$order <- c(1:5)

 #When we plot, we see line crossings
 plot(hc.tree)

 #Compute a correct ordering and replot
 element.order <- computeHclustPlotOrder(hc.tree$merge)
 hc.tree$order <- element.order
 plot(hc.tree)
```

EnsemblStyle *EnsemblStyle*

Description

Set the style of the chromosome names to Ensembl ("chr1" instead of "1")

Usage

```
EnsemblStyle(x)
```

Arguments

x	(GRanges, GRangesList or list of GRanges) The object to transform to Ensembl style

Value

The same x object with the styles of the seqlevels set to Ensembl

Examples

```
#GRanges
seg.data <- regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,8014630,1452
seg.data <- EnsemblStyle(seg.data)

#List of GRanges
seg.data <- list(a = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,801
             b=regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,8014630,
seg.data <- EnsemblStyle(seg.data)

#GRangesList
seg.data <- GRangesList(a = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50
                b = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,8014
seg.data <- EnsemblStyle(seg.data)
```

getBAFColumn *getBAFColumn*

Description

Identify the column in a data.frame or equivalent with the B-allele frecuency information

Usage

```
getBAFColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the bi-allelic frecuency information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getBAFColumn(df = df)
col.num <- getBAFColumn(df = df, col = "BAF")
```

---

getChrColumn *getChrColumn*

Description

Identify the column in a data.frame or equivalent with the chromosome information

Usage

```
getChrColumn(df, col=NULL, avoid.pattern = NULL, needed=TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the chromosome information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getChrColumn(df = df)
col.num <- getChrColumn(df = df, col = "chromosome")
```

---

getColumn *getColumn*

Description

Use simple pattern matching to try to identify a column in a data.frame or equivalent by its name

Usage

```
getColumn(df, col=NULL, pattern="", avoid.pattern = NULL, msg.col.name="", needed=TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
pattern	(character) The pattern to match the column name. If more than one column matches the pattern, the first one (leftmost) will be returned. The pattern may be any valid regular expression. (defaults to "")
avoid.pattern	(character) The pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
msg.col.name	(character) Only used in the error message to make the message clearer. The name of the column we are searching for. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will use pattern matching to try to identify which column of a data.frame contains a certain data. It will return its number. If `col` is specified it will not use pattern matching but identify if a column with that exact name exists and return its position.

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234", "endogenous" = FALSE, "chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5,  "strange.name"="strange.value")

col.num <- getColumn(df = df, pattern = "Chr|chr",  msg.col.name = "Chromosome", needed = TRUE)
col.num <- getColumn(df = df, col = "chromosome",  msg.col.name = "Chromosome", needed = TRUE)
col.num <- getColumn(df = df, col = 3, msg.col.name = "Chromosome", needed = TRUE)


col.num <- getColumn(df = df, pattern = "end",  msg.col.name = "End", needed = TRUE)
col.num <- getColumn(df = df, pattern = "end", avoid.pattern = "endogenous", msg.col.name = "End", needed = TRUE)
```

getCopyNumberColors *getCopyNumberColors*

Description

Get the colors representing the copy number levels

Usage

```
getCopyNumberColors(colors=NULL)
```

Arguments

colors	(character) Either a palette name (i.e. "green_orange_red") or a vector of colors. If NULL the default palette ("green_orange_red") will be returned. (defaults to NULL)

Details

This function returns a vector of colors with names from 0 to the length of the vector. The colors will be used to represent copy numbers from 0 (homozygous deletion) to the copy number equal to the length of the vector (typically 7 or 8). The function has a single parameter ‘colors‘, which can be either the name of one of the available palettes or the a vector of valid color specifications (i.e. c("red", "#FFAAAA", "gray", "#AAAAFF", "#5555FF", "#0000FF")). If ‘colors‘ is a palette name, the palette will be returned. If color is a color vector, it will be returned with the names set as needed.

Available palettes are: "gree_orange_red" and "red_blue"

Value

A vector of colors with the names set to the positions in the vector.

Examples

```
getCopyNumberColors()

getCopyNumberColors("red_blue")

getCopyNumberColors(c("red", "#FFAAAA", "gray", "#AAAAFF", "#5555FF", "#0000FF"))
```

getCopyNumberColumn *getCopyNumberColumn*

Description

Identify the column in a data.frame or equivalent with the copy number information

Usage

```
getCopyNumberColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the copy number information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getCopyNumberColumn(df = df)
col.num <- getCopyNumberColumn(df = df, col = "copy.number.level")
```

getEndColumn *getEndColumn*

Description

Identify the column in a data.frame or equivalent with the end position information

Usage

```
getEndColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the position information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getEndColumn(df = df)
col.num <- getEndColumn(df = df, col = "end.position")
```

getIDColumn *getIDColumn*

Description

Identify the column in a data.frame or equivalent with the ID information

Usage

```
getIDColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains ID information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
 col.num <- getIDColumn(df = df)
 col.num <- getIDColumn(df = df, col = "id")
```

getLOHColumn *getLOHColumn*

Description

Identify the column in a data.frame or equivalent with LOH information

Usage

```
getLOHColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the LOH information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getLOHColumn(df = df)
col.num <- getLOHColumn(df = df, col = "LOH")
```

getLRRColumn *getLRRColumn*

Description

Identify the column in a data.frame or equivalent with the Log Ratio information

Usage

```
getLRRColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the Log Ratio information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
 col.num <- getLRRColumn(df = df)
 col.num <- getLRRColumn(df = df, col = "Log.Ratio")
```

getPosColumn *getPosColumn*

Description

Identify the column in a data.frame or equivalent with the position information

Usage

```
getPosColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the position information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getPosColumn(df = df)
col.num <- getPosColumn(df = df, col = "strange.name")
```

getSegmentValueColumn *getSegmentValueColumn*

Description

Identify the column in a data.frame or equivalent with the position of the segment value information.

Usage

```
getSegmentValueColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame or equivalent that contains the segment information and return its position.

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getSegmentValueColumn(df = df)
col.num <- getSegmentValueColumn(df = df, col = "median.value.per.segment")
```

getStartColumn *getStartColumn*

Description

Identify the column in a data.frame with the start position information

Usage

```
getStartColumn(df, col = NULL, avoid.pattern = NULL, needed = TRUE, verbose = TRUE)
```

Arguments

df	(data.frame or equivalent) The object were columns are searched. It must be either a data.frame or an object where names(df) works.
col	(number or character) The column to identify. If NULL an heuristic will be used to automatically identify the column. It can also a number or a character to define the exact column to return.(defaults to NULL)
avoid.pattern	(character) An optional pattern to avoid on the column name. The pattern may be any valid regular expression. (defaults to "")
needed	(logical) Whether the column is needed or not. If TRUE, an error will be raised if the column is not found. (defaults to TRUE)
verbose	Whether to show information messages. (defaults to TRUE)

Details

Identify the column of a data.frame that contains the position information and return its position

Value

The number of the column matching the specification or NULL if no column was found.

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")
col.num <- getStartColumn(df = df )
col.num <- getStartColumn(df = df, col = "Start")
```

loadCopyNumberCalls *loadCopyNumberCalls*

Description

Loads copy number calls from a tabular format

Usage

```
loadCopyNumberCalls(cnv.data, chr.col = NULL, start.col = NULL, end.col = NULL, cn.col = NULL, loh.col =
```

Arguments

cnv.data	Either the name of the file with the data or a variable containing the data
chr.col	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to NULL)
start.col	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to NULL)
end.col	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to NULL)
cn.col	(number or character) The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to NULL)
loh.col	(number or character) The name or number of the column with LOH information. If NULL, it is automatically identified. (defaults to NULL)
segment.value.col	(number or character) The name or number of the column with segment value. If NULL, it is automatically identified. (defaults to NULL)
genome	(character) The name of the genome (defaults to NULL)
zero.based	(logical) Whether the data is zero-based and half open (i.e. ranges are defined by (start:end] so chr1:10-20 represents nine bases long features spanning from base 11 to 20). (defaults to FALSE)
verbose	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will load segments data from any "bed-like" data structure in R or file. Internally it uses the toGRanges function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. If no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment

Examples

```
df <- data.frame("id"= "rs1234","chromosome"="chr1", "Start"=0, "end.position"=100,
"copy.number.level"=3, "LOH"=0, "median.value.per.segment"=1.2,
"BAF"=0.2, "Log Ratio"=1.5, "strange.name"="strange.value")

cnv.call <- loadCopyNumberCalls(cnv.data = df)
```

`loadCopyNumberCallsCnmops` *loadCopyNumberCallsCnmops*

Description

Loads copy number calls from either cn.mops result class or in a tabular format

Usage

```
loadCopyNumberCallsCnmops(cn.mops.res, chr.col = NULL, start.col = NULL, end.col = NULL, cn.col = "CN",
```

Arguments

`cn.mops.res`	The name of the file with the data or the name of the variable with the data
`chr.col`	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to NULL)
`start.col`	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to NULL)
`end.col`	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to NULL)
`cn.col`	(number or character) The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to "CN")
`segment.value.col`	(number or character) The name or number of the column with segment value. It can be mean or median. If NULL, it is automatically identified. (defaults to "median")
`genome`	(character) The name of the genome (defaults to NULL)
`verbose`	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will load segments data from either cn.mops result class or in a tabular format from a file. Internally it uses the `toGRanges` function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. Column names of the format loaded are specified as default but, if no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment or a list of GRanges with a GRanges per sample.

Examples

```
# loadCopyNumberCallsCnmops from cn.mops result class:
#NOT RUN - Example loading data from a results object created by cn.mops
# require(cn.mops)
# data(cn.mops, package = "cn.mops")

# cn.mops.res <- cn.mops(XRanges)
# cn.mops.res <- calcIntegerCopyNumbers(cn.mops.res)
# cnv.call <- loadCopyNumberCallsCnmops(cn.mops.res = cn.mops.res)

# loadCopyNumberCallsCnmops from a file where cn.mops result was saved:
cn.mops.res <- system.file("extdata", "cn.mops.segmentation.csv", package = "CopyNumberPlots", mustWork = TRUE)
cnv.call <- loadCopyNumberCallsCnmops(cn.mops.res = cn.mops.res)
```

---

loadCopyNumberCallsCNVkit

loadCopyNumberCallsCNVkit

Description

Loads copy number calls from CNVkit.cns file format

Usage

```
loadCopyNumberCallsCNVkit(cnvkit.file, chr.col = "chromosome", start.col = "start", end.col = "end", se
```

Arguments

cnvkit.file	The name of the file with the data
chr.col	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to "chromosome")
start.col	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to "start")
end.col	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to "end")
segment.value.col	(number or character) The name or number of the column with segment value. If NULL, it is automatically identified. (defaults to "log2")
file.type	(character) wheter to load ".cns", ".cnr" or "auto" if the file type is automatically recognised. (defaults to "auto)

cn.col	(number or character) The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to NULL)
zero.based	(logical) Whether the data is zero-based and half open (i.e. ranges are defined by (start:end] so chr1:10-20 represents nine bases long features spanning from base 11 to 20). (defaults to FALSE)
genome	(character) The name of the genome (defaults to NULL)
verbose	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will load segments data from CNVkit.cns file format and from CNVkit.cnr. Internally it uses the `toGRanges` function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. If no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment.

Examples

```
## loadCopyNumberCallsCNVkit from .cns file format:
## An example of .cns file format is found at https://github.com/etal/cnvkit/blob/master/test/formats/cl_seq.cns.

# Loading a CNVkit.cns file
cnvkit.file <- system.file("extdata", "CNVkit_output.cns", package = "CopyNumberPlots", mustWork = TRUE)
cnv.call <- loadCopyNumberCallsCNVkit(cnvkit.file)

# Loading a CNVkit.cnr file
cnvkit.file <- system.file("extdata", "CNVkit_output.cnr", package = "CopyNumberPlots", mustWork = TRUE)
cnv.call <- loadCopyNumberCallsCNVkit(cnvkit.file)
```

`loadCopyNumberCallsDECoN` *loadCopyNumberCallsDECoN*

Description

Loads copy number calls from DECoN output file

Usage

```
loadCopyNumberCallsDECoN(decon.file, chr.col = NULL, start.col = "Start", end.col = "End", cn.col = NUL
```

Arguments

decon.file	The name of the file with the data
chr.col	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to NULL)
start.col	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to "Start")
end.col	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to "End")
cn.col	(number or character) The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to NA)
segment.value.col	(number or character) The name or number of the column with segment value. If NULL, it is automatically identified. (defaults to NULL)
genome	(character) The name of the genome. (defaults to NULL)
verbose	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will load segments data from DECoN output file. Internally it uses the toGRanges function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. If no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment or a list of GRanges with a GRanges per sample.

Examples

```
decon.file <- system.file("extdata", "DECoN_output.txt", package = "CopyNumberPlots", mustWork = TRUE)
cn.calls <- loadCopyNumberCallsDECoN(decon.file = decon.file)
```

---

loadCopyNumberCallsDNAcopy

loadCopyNumberCallsDNAcopy

Description

Loads copy number calls from DNACopy results.

Usage

```
loadCopyNumberCallsDNAcopy(DNAcopy.data, chr.col = "chrom", start.col = "loc.start", end.col = "loc.en
```

Arguments

DNAcopy.data	The name of the file with the data
chr.col	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to "chrom")
start.col	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to "loc.start")
end.col	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to "loc.end")
segment.value.col	(number or character) The name or number of the column with segment value. If NULL, it is automatically identified. (defaults to "seg.mean")
cn.col	(number or character) The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to NULL)
chr.transformation	(character) The transformation of the chromosome names in a comma separated "key:value" format.(defaults to "23:X,24:Y,25:MT")
genome	(character) The name of the genome (defaults to NULL)
verbose	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will load segments data from DNAcopy data structure in R. Internally it uses the toGRanges function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. If no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment or a list of GRanges with a GRanges per sample.

Examples

```
library(DNAcopy)

data(coriell)
CNA.object <- suppressWarnings(CNA(cbind(coriell$Coriell.05296), coriell$Chromosome, coriell$Position, data.type

smoothed.CNA.object <- smooth.CNA(CNA.object)
DNAcopy.data <- segment(smoothed.CNA.object, verbose=1)

cnv.call <- loadCopyNumberCallsDNAcopy(DNAcopy.data = DNAcopy.data)

# more than 1 sample
CNA.object <- CNA(genomdat = cbind(coriell$Coriell.05296, coriell$Coriell.13330), chrom = coriell$Chromosome, map
smoothed.CNA.object <- smooth.CNA(CNA.object)
DNAcopy.data <- segment(smoothed.CNA.object, verbose=1)

cnv.call <- loadCopyNumberCallsDNAcopy(DNAcopy.data = DNAcopy.data)
```

loadCopyNumberCallsPanelcnmops

loadCopyNumberCallsPanelcnmops

Description

Loads copy number calls from either cn.mops result class or in a tabular format

Usage

```
loadCopyNumberCallsPanelcnmops(panelcn.mops.res, chr.col = NULL, start.col = NULL, end.col = NULL, cn.c
```

Arguments

panelcn.mops.res	The name of the file with the data or the name of the variable with the data
chr.col	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to NULL)
start.col	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to NULL)
end.col	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to NULL)
cn.col	(number or character) The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to NULL)
segment.value.col	(number or character) The name or number of the column with segment value. If NULL, it is automatically identified. (defaults to NULL)
genome	(character) The name of the genome (defaults to NULL)
verbose	(character) Whether to show information messages. (defaults to TRUE)

Details

This function will load segments data from panelcn.mops resulttable or in a tabular format or file. Internally it uses the `toGRanges` function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. If no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment or a list of GRanges with a GRanges per sample.

Examples

```
## loadCopyNumberCallsPanelcnmops from panelcn.mops resulttable:
library(panelcn.mops)
data(panelcn.mops, package = "panelcn.mops")
XandCB <- test
sampleNames <- colnames(elementMetadata(XandCB))
elementMetadata(XandCB) <- cbind(elementMetadata(XandCB), elementMetadata(control))

resulttable <- createResultTable(resultlist = resultlist, XandCB = XandCB, countWindows = countWindows, sampleName
panelcn.mops.res <- resulttable[[1]]
cnv.call <- loadCopyNumberCallsPanelcnmops(panelcn.mops.res = panelcn.mops.res)
```

`loadCopyNumberCallspennCNV`

loadCopyNumberCallspennCNV

Description

Loads copy number calls from pennCNV.rawcnv file format

Usage

```
loadCopyNumberCallspennCNV (pennCNV.file, chr.col = NULL, start.col = NULL, end.col = NULL, cn.col = NUL
```

Arguments

`pennCNV.file`	The name of the file with the data
`chr.col`	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to NULL)
`start.col`	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to NULL)
`end.col`	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to NULL)
`cn.col`	(number or character) The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to NULL)
`segment.value.col`	(number or character) The name or number of the column with segment value. If NULL, it is automatically identified. (defaults to NULL)
`genome`	(character) The name of the genome (defaults to NULL)
`verbose`	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will load copy number calls from pennCNV.rawcnv file format. Internally it uses the `toGRanges` function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. If no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment or a list of GRanges with a GRanges per sample.

Examples

```
## loadCopyNumberCallspennCNV from .rawcnv file format:
## The file to run the example can be found in: http://penncnv.openbioinformatics.org/en/latest/user-guide/test/

pennCNV.file <- system.file("extdata", "pennCNV.rawcnv", package = "CopyNumberPlots", mustWork = TRUE)
cnv.call <- loadCopyNumberCallspennCNV(pennCNV.file = pennCNV.file)
```

---

loadCopyNumberCallsSeg

loadCopyNumberCallsSeg

Description

Loads copy number calls from .seg file format

Usage

```
loadCopyNumberCallsSeg (seg.file, chr.col = "chrom", start.col = "loc.start", end.col = "loc.end", segm
```

Arguments

seg.file	The name of the file with the data
chr.col	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to "chrom")
start.col	(number or character) The name or number of the column with start position information. If NULL, it is automatically identified. (defaults to "loc.start")
end.col	(number or character) The name or number of the column with end position information. If NULL, it is automatically identified. (defaults to "loc.end")
segment.value.col	(number or character) The name or number of the column with segment value. If NULL, it is automatically identified. (defaults to "seg.mean")
cn.col	(number or character)The name or number of the column with CN information. If NULL, it is automatically identified. (defaults to NULL)
genome	(character) The name of the genome (defaults to NULL)
chr.transformation	(character)(character) The transformation of the chromosome names in a comma separated "key:value" format as detailed at https://cnvkit.readthedocs.io/en/stable/importexport.html#impo seg.(defaults to "23:X,24:Y,25:MT")
verbose	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will load segments data from .seg file format. Internally it uses the `toGRanges` function from regioneR package and can work with any format accepted by it, including R objects and local or remote files. If no column names are specified, it will use simple heuristics to try to identify the relevant data columns.

Value

A GRanges with a range per copy number segment

Examples

```
## loadCopyNumberCallsSeg from .seg file format:
## the file to run in the example can be found in:
## https://software.broadinstitute.org/software/igv/SEG
## under example.seg file name.

seg.file <- system.file("extdata", "DNACopy_output.seg", package = "CopyNumberPlots", mustWork = TRUE)
cnv.call <- loadCopyNumberCallsSeg(seg.file = seg.file)
```

`loadSNPData` *loadSNPData*

Description

Loads SNP array data in a tabular format

Usage

```
loadSNPData(snp.data, chr.col = NULL, start.col = NULL, end.col = NULL, pos.col = NULL, baf.col = NULL, l
```

Arguments

`snp.data`	Either the name of the file with the data or a variable containing the data.
`chr.col`	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to NULL)
`start.col`	(number or character) the name or number of the column with start position
`end.col`	(number or character) the name of number of the column with end position
`pos.col`	(number or character) The name or number of the column with position information. If NULL, it is automatically identified. (defaults to NULL)
`baf.col`	(number or character) The name or number of the column with BAF information. If NULL, it is automatically identified. (defaults to NULL)
`lrr.col`	(number or character) The name or number of the column with LRR information. If NULL, it is automatically identified. (defaults to NULL)

id.col	(number or character) The name or number of the column with SNP identifier information. If NULL, it is automatically identified. (defaults to NULL)
genome	(character) The name of the genome (defaults to "hg19")
verbose	Wether information messages should be generated. (defaults to TRUE)

Details

Given a file name or data in a tabular format, the function loads SNP array data in a tabular format. It will try to identify the columns with the relevant information (chr, position, BAF, LRR, etc...) or will use the column number or name supplied by the user, if any. It will convert the tabular data into a GRanges, with one range per SNP in the table.

Value

A GRanges with a range per SNP

Examples

```
# There are two examples of posible files to load.
snp.data1 <- system.file("extdata", "snp.data_test.csv", package = "CopyNumberPlots", mustWork=TRUE)
snps <- loadSNPData(snp.data = snp.data1)

snp.data2 <- system.file("extdata", "snp.data_test2.csv", package = "CopyNumberPlots", mustWork=TRUE)
snps <- loadSNPData(snp.data = snp.data2)
```

---

loadSNPDataDNAcopy *loadSNPDataDNAcopy*

Description

Loads SNP array data in a tabular format using data from DNAcopy rawdata or DNAcopy results

Usage

```
loadSNPDataDNAcopy (snp.data, chr.col = "chrom", start.col = NULL, end.col = NULL, pos.col = "maploc", l
```

Arguments

snp.data	Either the name of the file with the data or a variable containing the data.
chr.col	(number or character) The name or number of the column with chromosome information. If NULL, it is automatically identified. (defaults to NULL)
start.col	(number or character) the name or number of the column with start position
end.col	(number or character) the name of number of the column with end position
pos.col	(number or character) The name or number of the column with position information. If NULL, it is automatically identified. (defaults to NULL)

lrr.col	(number or character) The name or number of the column with LRR information. If NULL, it is automatically identified. (defaults to NULL)
chr.transformation	(character) The transformation of the chromosome names in a comma separated "key:value" format.(defaults to "23:X,24:Y,25:MT")
genome	(character) The name of the genome (defaults to "hg19")
na.rm	(logical) Whether to remove NA values. (defaults to FALSE)
verbose	Wether information messages should be generated. (defaults to TRUE)

Details

Given a DNAcopy rawdata or DNAcopy results object, the function load SNP array data in a tabular format. It will try to identify the columns with the relevant information (chr, position, BAF, LRR, etc...) or will use the column number or name supplied by the user, if any. It will convert the tabular data into a GRanges, with one range per SNP in the table.

Value

A list of GRanges with a range per SNP and one GRanges per sample.

Examples

```
library(DNAcopy)
data(coriell)
#one sample
CNA.object <- CNA(cbind(coriell$Coriell.05296), coriell$Chromosome, coriell$Position, data.type="logratio")
CNA.object <-  smooth.CNA(CNA.object)
snps <- loadSNPDataDNAcopy(snp.data = CNA.object, na.rm = TRUE)

# more than 1 sample
CNA.object <- CNA(genomdat = cbind(coriell$Coriell.05296, coriell$Coriell.13330), chrom = coriell$Chromosome, map
CNA.object <-  smooth.CNA(CNA.object)
snps <- loadSNPDataDNAcopy(snp.data = CNA.object, na.rm = TRUE)

#If the data come from DNAcopy results
DNAcopy.object <- segment(CNA.object, verbose=1)
snps <- loadSNPDataDNAcopy(snp.data = DNAcopy.object)
```

---

loadSNPDataFromVCF *loadSNPDataFromVCF*

Description

Load BAF-like data from a VCF file based on the variant allele frequency

Usage

```
loadSNPDataFromVCF(vcf.file, regions=NULL, genome="hg19", mirror.baf=TRUE, verbose=TRUE)
```

Arguments

vcf.file	The name of the VCF file with the data
regions	The regions to which we will limit the import (defaults to NULL)
genome	(a character)The name of the genome (defaults to "hg19")
mirror.baf	Flip the baf of about half the snps (the ones in odd positions in the genome) to achieve a mirror-like effect as in SNP arrays (defaults to TRUE)
verbose	Wether information messages should be generated. (defaults to TRUE)

Details

Given a VCF file the function will compute BAF-like values based on the variant allele frequency contained in it and return a GRanges object with a column named "baf".

Value

A GRanges object with a range per variant and a column named baf

Examples

```
vcf.file <- system.file("extdata", "example.vcf.gz", package = "CopyNumberPlots", mustWork = TRUE)
# These examples fail in windows. Commenting them out temporarily
# snps <- loadSNPDataFromVCF(vcf.file)

#kp <- plotKaryotype(plot.type = 4)
#plotBAF(kp, snps = snps, labels = names(snps))
```

---

plotBAF *plotBAF*

Description

Plot the B-allele frequency (BAF) data

Usage

```
plotBAF(karyoplot, snps, baf.column="baf", labels=NULL, points.cex=0.3, points.col="#333333", points.
```

Arguments

karyoplot	(a KaryoPlot object) The object returned by the plotKaryotype function and representing the current active plot.
snps	(a GRanges, a list of GRanges or a GRangesList) An object with the positions of the SNPs and a column with the BAF values. Other columns are ignored. If it's a list of GRanges with different samples, all samples will be plotted, splitting the total plot space between them.

baf.column	(a character) The name of the column in snps with the BAF values. (defaults to "baf")
labels	(a character) The text of the label to identify the data. If NA, no label will be plotted. If NULL, if snps is a single sample GRanges it will default to "BAF", if it's a list of samples it will default to the names in the list or consecutive numbers if names(snps) is NULL. (defaults to NULL)
points.cex	(numeric) The size of the points. (defaults to 0.3)
points.col	(a color) The color of the points (defaults to "#333333")
points.pch	(numeric) The shape of the points. (defaults to 16, a filled circle)
label.cex	(numeric) The size of the label (defaults to 1.5)
label.srt	(numeric) The rotation of the label (defaults to 90, vertical text)
label.margin	(numeric) The margin between the label and the origin of the plot in plot coordinates (the width of the plot is 1). (defaults to 0.03)
add.axis	(logical) Whether to add an axis (defaults to TRUE)
axis.cex	(numeric) The size of the axis labels. (defaults to 1.2)
r0	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)
r1	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)
track.margin	(numeric) If snps is a list object, this is the margin between the samples BAF. (deafults to 0.1)
data.panel	(numeric) (karyoploteR parameter) The identifier of the data panel where the data is to be plotted. The available data panels depend on the plot type selected in the call to plotKaryotype. (defaults to 1)
verbose	(logical) Wether messages with information on the processing should be generated (defaults to FALSE)
...	The ellipsis operator can be used to specify any additional graphical parameters. Any additional parameter will be passed to the internal calls to karyoploteR functions.

Details

This function plot the B-allele frequency (BAF) values on the genome. BAF values represent the frequency of one of the alleles (NOT always the minor allele) in the sample and usually come from SNP-array. However, it's possible to get similar values from NGS variant calling data (loadSNPDataFromVCF for example, will load a standard VCF and generate BAF-like values). The function plots the data points, an optional axis and an optional label identifying the data. The function expects a GRanges object with the position of the SNPs with a column (usually named 'baf') with the BAF values or any object valid to toGRanges. This object can be created with loadSNPData.

Value

Invisibly returns the karyoplot object representing the plot. With it it is possible to add other elements to the plot using standrad karyoploteR functions

Examples

```
pos <- sort(floor(runif(1000, 1, 10000)))
baf.data <- toGRanges("chr1", pos, pos)
baf.data$baf <- rnorm(1000, mean = 0.5, sd = 0.05)
baf.data$baf[1:400] <- baf.data$baf[1:400] + c(0.2, -0.2)

kp <- plotKaryotype(zoom=toGRanges("chr1", 1, 10000))
plotBAF(kp, baf.data)


names(mcols(baf.data)) <- "values"
kp <- plotKaryotype(zoom=toGRanges("chr1", 1, 10000))
plotBAF(kp, baf.data, baf.column="values", r0=0, r1=0.5, points.col="red", labels="Values")
plotBAF(kp, baf.data, baf.column=1, r0=0.5, r1=1, points.col="orange", labels="First", label.cex=0.8, points.cex=

#Plotting a list
baf.data2 <- baf.data
baf.data2$baf <- rnorm(1000, mean = 0.5, sd = 0.05)
baf.data$baf <- rnorm(1000, mean = 0.5, sd = 0.05)
baf.data$baf[1:400] <- baf.data$baf[1:400] + c(0.2, -0.2)

baf.list <- list(Tumor=baf.data, Normal=baf.data2)


kp <- plotKaryotype(zoom=toGRanges("chr1", 1, 10000))
plotBAF(kp, baf.list, labels=NULL)

kp <- plotKaryotype(zoom=toGRanges("chr1", 1, 10000))
plotBAF(kp, baf.list, labels=NULL, r1=0.8, label.cex=0.8, axis.cex=0.4)
kpRect(kp, chr="chr1", x0=0, x1=pos[400], y0=0, y1=0.8, border="red")
kpPlotNames(kp, chr="chr1", x0=0, x1=pos[400], y0=0, y1=0.8, labels = "Alteration", position = "top", col="red")
```

---

plotCopyNumberCalls *plotCopyNumberCalls*

Description

Plot the segments representing the copy number calls by any algorithm

Usage

```
plotCopyNumberCalls(karyoplot, cn.calls, cn.values=NULL, cn.column="cn", cn.colors=NULL, loh.values=N
```

Arguments

karyoplot	A karyoplote object
cn.calls	(a GRanges, a list of GRanges or a GRangesList) An object with the positions of the CN calls and a column with the CN values. Other columns are ignored. If it's a list of GRanges with different samples, all samples will be plotted, splitting the total plot space between them.
cn.values	(integer vector) The CN values. If NULL, they will be extracted from the cn.calls (defaults to NULL)
cn.column	(integer or character vector) The name or number of the column with CN information.(defaults to "cn")
cn.colors	(colors) The colors assigned to gains and losses (defaults to NULL)
loh.values	(logical vector) A logical vector that indicates whether there is or not LOH or a vector that can be coerced as logical. (defaults to NULL)
loh.column	(number or character) The name or number of the column with LRR information. (defaults to "loh")
loh.color	(a color) The color assigned to LOH values. (defaults to "#1E90FF")
loh.height	(numeric) The proportion of r0 and r1 of the vertical space over each chromosome dedicated to loh. It is dedicated the 30% of the vertical space by default.(defaults to 0.3)
labels	(character) The text of the label to identify the data. If NA, no label will be plotted. If NULL, if snps is a single sample GRanges it will default to "CN", if it's a list of samples it will default to the names in the list or consecutive numbers if names(snps) is NULL. (defaults to NULL)
label.cex	(numeric) The size of tthe label (defaults to 1)
label2.cex	(numeric) The size of the label 2. If NULL label2.cex will be lable.cex. (defaults to NULL)
track.margin	(numeric) If cn.calls is a list object, this is the margin between the samples CN. (deafults to 0.01)
r0	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)(defaults to 0)
r1	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)(defaults to 1)
...	The ellipsis operator can be used to specify any additional graphical parameters. Any additional parameter will be passed to the internal calls to karyoploteR functions.

Details

Plots the copy number calls as colored rectangles in the karyoplot. Each copy number status has its own color (including 2n, normal genome) and these colors may be specified by cn.colors. If LOH data is available it will be plotted below the copy number data, with a colored rectangle for every LOH region. The input is simply a GRanges object with an additional column containing the copy number status (as integers) and optionally another column containing the LOH status of each region.

If there's only one label, it will be used to label both Copy Number and LOH. If there are at least two labels, the first one is used to label the Copy Number and the second one to labvel the LOH.

If the function is called with a list of GRanges it plots every GRanges independently as different tracks one below the other. Track positioning is based on 'autotrack' and the margin between track is controlled by 'track.margin'.

Value

Invisibly returns the karyoplot object representing the plot. With it it is possible to add other elements to the plot using standrad karyoploteR functions

Examples

```
s1.calls.file <- system.file("extdata", "S1.segments.txt", package = "CopyNumberPlots", mustWork = TRUE)
s1.calls <- loadCopyNumberCalls(s1.calls.file)

s2.calls.file <- system.file("extdata", "S2.segments.txt", package = "CopyNumberPlots", mustWork = TRUE)
s2.calls <- loadCopyNumberCalls(s2.calls.file)

kp <- plotKaryotype(chromosomes="chr1")
#plotCopyNumberCalls(kp, s1.calls)

kp <- plotKaryotype(chromosomes="chr1")
#plotCopyNumberCalls(kp, s1.calls, cn.colors="red_blue")

#List of GRanges
cn.calls <- list(s1=s1.calls, s2 =s2.calls)

kp <-plotKaryotype(chromosomes="chr1")
#plotCopyNumberCalls(kp, cn.calls, cn.colors="red_blue")
```

`plotCopyNumberCallsAsLines`

plotCopyNumberCallsAsLines

Description

Plot the segments representing the copy number calls by any algorithm

Usage

```
plotCopyNumberCallsAsLines(karyoplot, cn.calls, style="line", cn.column="cn", labels=NULL, label.cex=
```

Arguments

karyoplot	(a KaryoPlot object) The object returned by the plotKaryotype function and representing the current active plot.
cn.calls	(a GRanges, a list of GRanges or a GRangesList) An object with the positions of the CN calls and a column with the CN values. Other columns are ignored. If it's a list of GRanges with different samples, all samples will be plotted, splitting the total plot space between them.
style	(character) The style in which the lines can be plot. It colud be as lines or segments. (defaults to "line")
cn.column	(integer or character vector) The name or number of the column with CN information.(defaults to "cn")
labels	(character) The text of the label to identify the data. If NA, no label will be plotted. If NULL, if snps is a single sample GRanges it will default to "Segments", if it's a list of samples it will default to the names in the list or consecutive numbers if names(snps) is NULL. (defaults to NULL)
label.cex	(numeric) The size of the label (defaults to 1)
add.axis	(logical) Whether to plot an axis (defaults to TRUE)
axis.cex	(numeric) The size of the axis labels.(defaults to 1)
numticks	The number of tocks in the axis (defaults to NULL)
col	(color) The color of the lines (defaults to "black")
ymin	(numeric) (karyoploteR parameter) The minimum value of y to be plotted. If NULL, it is set to the min value of the selected data panel. (defaults to -4)
ymax	(numeric) (karyoploteR parameter) (numeric) The maximum value of y to be plotted. If NULL, it is set to the max value of the selected data panel. (defaults to 2)
r0	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to 0)
r1	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to 1)
...	The ellipsis operator can be used to specify any additional graphical parameters. Any additional parameter will be passed to the internal calls to karyoploteR functions.

Details

Plots the segments

Value

Invisibly returns the karyoplot object representing the plot. With it it is possible to add other elements to the plot using standrad karyoploteR functions

Examples

```
cncalls.file <- system.file("extdata", "S1.segments.txt", package = "CopyNumberPlots", mustWork = TRUE)
cncalls <- loadCopyNumberCalls(cncalls.file)

kp <- plotKaryotype("hg19")
#plotCopyNumberCallsAsLines(kp, cncalls)

kp <- plotKaryotype("hg19", chromosomes="chr3")
#plotCopyNumberCalls(kp, cncalls, r0=0, r1=0.2)
#plotCopyNumberCallsAsLines(kp, cncalls, r0=0.25, r1=0.55)
#plotCopyNumberCallsAsLines(kp, cncalls, r0=0.65, r1=0.95, style="segments", col="red", ymin=0, ymax=4, numticks=

kp <- plotKaryotype("hg19")
#plotCopyNumberCallsAsLines(kp, cncalls, r0=0.2, r1=0.4, style="segments", add.axis=FALSE, labels="CopyNumber")

#List of GRanges
cncalls.list <- list(s1=cncalls, s2 =cncalls)
kp <- plotKaryotype("hg19")
#plotCopyNumberCallsAsLines(kp, cncalls.list,style="segments", add.axis=FALSE)
```

---

plotCopyNumberSummary *plotCopyNumberSummary*

Description

Plot a summary of copy number status over a number of samples using a histogram-like representation.

Usage

```
plotCopyNumberSummary(karyoplot, cn.calls, direction="in", gain.color=NULL, normal.color=NULL, loss.
```

Arguments

karyoplot	(a KaryoPlot object) The object returned by the plotKaryotype function and representing the current active plot.
cn.calls	(a list of GRanges or a GRangesList) A list of GRanges or a GRangesList containing the GRanges objects with cn.column value

direction	The direction to which the coverage plot point, either "in" for inward or "out" for outward. (defaults to "in")
gain.color	(color) The color assigned to gains (defaults to NULL)
normal.color	(color) The color assigned to normal ploidy(defaults to NULL)
loss.color	(colors) The color assigned to losses(defaults to NULL)
add.grid	(logical) Whether to add lines as a grid in the plot (defaults to FALSE)
grid.color	(color) The color of the grid (defaults to "white")
labels	(a character) The text of the label to identify the data. If NA, no label will be plotted. If NULL, if snps is a single sample GRanges it will default to "CN", if it's a list of samples it will default to the names in the list or consecutive numbers if names(snps) is NULL. (defaults to NULL)
label.cex	(numeric) The size of the label (defaults to 1.5)
label.srt	(numeric) The rotation of the label (defaults to 90, vertical text)
pos	The position of the label (defaults to 2)
r0	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)
r1	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)
...	The ellipsis operator can be used to specify any additional graphical parameters. Any additional parameter will be passed to the internal calls to karyoploteR functions.

Details

2 types of plots. Convergent or divergent barplots

Value

Invisibly returns the karyoplot object representing the plot. With it it is possible to add other elements to the plot using standrad karyoploteR functions

Examples

```
all.scnas <- list(
        loadCopyNumberCalls(system.file("extdata", "S1.segments.txt", package = "CopyNumberPlots", mustWork = TR
        loadCopyNumberCalls(system.file("extdata", "S2.segments.txt", package = "CopyNumberPlots", mustWork = TR
        loadCopyNumberCalls(system.file("extdata", "S3.segments.txt", package = "CopyNumberPlots", mustWork = TR
            )

kp <- plotKaryotype(chromosomes="chr1")
```

```
#plotCopyNumberCalls(kp, all.scnas, r0=0.4, r1=1)
#plotCopyNumberSummary(kp, all.scnas, r0=0, r1=0.35)

kp <- plotKaryotype(chromosomes="chr1")
#plotCopyNumberSummary(kp, all.scnas, gain.col="blue", loss.col="red", labels="Copy Number", r0=0, r1=0.3, label.
#plotCopyNumberSummary(kp, all.scnas, direction="out", add.grid=TRUE, r0=0.35, r1=0.65)
#plotCopyNumberSummary(kp, all.scnas, direction="out", r0=0.7, r1=1)
kpAxis(kp, ymin=0, ymax=10, r0=0.85, r1=1, tick.pos = c(0,5,10))
kpAxis(kp, ymin=0, ymax=10, r0=0.85, r1=0.7, tick.pos = c(5,10))


#NOT RUN (time constraints in Bioconductor checks):
# Use a random example with more realistic results
# gg <- filterChromosomes(getGenome("hg19"))

# all.scnas <- list()
# for(i in seq_len(10)) {
#   scnas <- createRandomRegions(40, 10e6, 10e6)
#   scnas$cn <- floor(runif(40, 0, 4))
#   scnas$loh <- ifelse(scnas$cn<2, 1, 0)
#   normal.regs <- subtractRegions(gg, scnas)
#   normal.regs$cn <- 2
#   scnas <- sort(c(scnas, normal.regs))
#   all.scnas[[paste0("samp", i)]] <- scnas
# }

# kp <- plotKaryotype("hg19", plot.type=4)
# plotCopyNumberSummary(kp, all.scnas)

# kp <- plotKaryotype("hg19", plot.type=4)
# plotCopyNumberSummary(kp, all.scnas, gain.col="blue", loss.col="red", labels="Copy Number", r0=0, r1=0.3, label
# plotCopyNumberSummary(kp, all.scnas, direction="out", add.grid=TRUE, r0=0.35, r1=0.65)
# plotCopyNumberSummary(kp, all.scnas, direction="out", r0=0.7, r1=1)
# kpAxis(kp, ymin=0, ymax=10, r0=0.85, r1=1, tick.pos = c(0,5,10))
# kpAxis(kp, ymin=0, ymax=10, r0=0.85, r1=0.7, tick.pos = c(5,10))
```

---

plotLRR *plotLRR*

Description

Plots the raw SNP array data using karyoploteR

Usage

```
plotLRR(karyoplot, snps, lrr.column="lrr", labels="LRR", ymin=-4, ymax=2, out.of.range = "points", out
```

Arguments

karyoplot	(a KaryoPlot object) The object returned by the plotKaryotype function and representing the current active plot.
snps	(a GRanges, a list of GRanges or a GRangesList) An object with the positions of the SNPs and a column with the LRR values. Other columns are ignored. If it's a list of GRanges with different samples, all samples will be plotted, splitting the total plot space between them.
lrr.column	(number or character) The name or number of the column with LRR information. (defaults to "lrr")
labels	(character) The text of the label to identify the data. If NA, no label will be plotted. If NULL, if snps is a single sample GRanges it will default to "LRR", if it's a list of samples it will default to the names in the list or consecutive numbers if names(snps) is NULL.(defaults to "LRR")
ymin	(numeric) (karyoploteR parameter) The minimum value of y to be plotted. If NULL, it is set to the min value of the selected data panel. (defaults to -4)
ymax	(numeric) (karyoploteR parameter) (numeric) The maximum value of y to be plotted. If NULL, it is set to the max value of the selected data panel. (defaults to 2)
out.of.range	(a character) Either to plot "points" or "density" (defaults to "points")
out.of.range.col	The color the out-of-range points should be plotted (defaults to "red")
density.height	The height of the maximum density peak if out.of.range is "density" (defaults to 0.05)
density.window	The window used to compute the uot-of-range density (defaults to 1e5)
line.at.0	(logical) Whether to plot an horizontal line at 0. (defaults to TRUE)
line.at.0.col	(color) The color of the horizontal line plotted at 0. (defaults to "blue")
r0	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)
r1	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)
points.cex	(numeric) The size of the points. (defaults to 0.3)
points.col	(a color) The color of the points (defaults to "#333333")
points.pch	(numeric) The shape of the points. (defaults to 16, a filled circle)
label.cex	(numeric) The size of the label (defaults to 1.5)
label.srt	(numeric) The rotation of the label (defaults to 90, vertical text)
label.margin	(numeric) The margin between the label and the origin of the plot in plot coordinates (the width of the plot is 1). (defaults to 0.03)

add.axis	(logical) Whether to add an axis (defaults to TRUE)
axis.cex	(numeric) The size of the axis labels. (defaults to 1.2)
track.margin	(numeric) If snps is a list object, this is the margin between the samples BAF. (deafults to 0.1)
data.panel	(numeric) (karyoploteR parameter) The identifier of the data panel where the data is to be plotted. The available data panels depend on the plot type selected in the call to plotKaryotype. (defaults to 1)
verbose	(logical) Wether messages with information on the processing should be generated (defaults to FALSE)
...	The ellipsis operator can be used to specify any additional graphical parameters. Any additional parameter will be passed to the internal calls to karyoploteR functions.

Details

Creates a plot with the LRR values along the genome

Value

Invisibly returns the karyoplot object representing the plot. With it it is possible to add other elements to the plot using standrad karyoploteR functions

Examples

```
pos <- floor(runif(1000, 1, 10000))
lrr.data <- toGRanges("chr1", pos, pos)
lrr.data$lrr <- rnorm(1000, mean = 0, sd = 0.3)

kp <- plotKaryotype(zoom=toGRanges("chr1", 1, 10000))
plotLRR(kp, lrr.data)


kp <- plotKaryotype(zoom=toGRanges("chr1", 1, 10000))
plotLRR(kp, lrr.data, lrr.column=1, points.col="orange", labels="First", label.cex=0.8, points.cex=1.4)
```

---

plotSingleCellCopyNumberCalls

plotSingleCellCopyNumberCalls

Description

Plot the segments representing the copy number calls from a single-cell CNV data file. The file must be an HDF5 file with the same format as the ones produced by 10X CellRanger software.

Usage

```
plotSingleCellCopyNumberCalls(karyoplot, cnv.file, reorder=TRUE, cn.colors=NULL, track.margin=0, r0=0
```

Arguments

karyoplot	A karyoplote object
cnv.file	(an HDF5 file) The path to an HDF5 file containing the single-cell CNV data.
reorder	(logical) If TRUE, the cells will be plotted inthe order specified by the hirearchical clustering tree embedded in the HDF5 file. If FALSE the cells will be plotted in the order they are stored in the file. (defaukts to TRUE)
cn.colors	(colors) The colors assigned to gains and losses. getCopyNumberColors is used to determine them. If NULL, the default color scheme is used. (defaults to NULL)
track.margin	(numeric) This is the margin between the cells CN, in portion of the total per cell space. (deafults to 0)
r0	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)(defaults to 0)
r1	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)(defaults to 1)
...	The ellipsis operator can be used to specify any additional graphical parameters. Any additional parameter will be passed to the internal calls to karyoploteR functions.

Details

This function will open the HDF5 file, extract the CNV values for each cell and plot them as a different colored segments using plotCopyNumberCalls. By default cells will be reordered and plotted according the hierarchical clustering tree embedded in the file. If reorder is FALSE or the tree is not present, cell will be plotted in the order they are stored in the file.

Value

Invisibly returns the karyoplot object representing the plot. With it it is possible to add other elements to the plot using standrad karyoploteR functions. The returned pbject will have an additional "latest.plot" element with a list containing: the number of cells, the hierarchical clustering tree, the bin size used to partition the genome, the windows representing such partitioning, the number of bins per chromosome and a GRAnges with the regions where no cell had any data, the no-call regions.

Note

If the file is open by any other application the function will fail.

Examples

```
kp <- plotKaryotype(plot.type=4, genome="hg38")
#NOT RUN - Using 10X example data from https://www.10xgenomics.com/resources/datasets/
#plotSingleCellCopyNumberCalls(kp, "breast_tissue_D_2k_cnv_data.h5")
```

`plotSingleCellCopyNumberSummary`
plotSingleCellCopyNumberSummary

Description

Plot a summary of the copy number calls from a single-cell CNV data file. Plot a histogram-like plot with the number of cells with gains and losses. The file must be an HDF5 file with the same format as the ones produced by 10X CellRanger software.

Usage

```
plotSingleCellCopyNumberSummary(karyoplot, cnv.file, direction="in", gain.color=NULL, normal.color=N
```

Arguments

`karyoplot`	A karyoplot object
`cnv.file`	(an HDF5 file) The path to an HDF5 file containing the single-cell CNV data.
`direction`	The direction to which the coverage plot point, either "in" for inward or "out" for outward. (defaults to "in")
`gain.color`	(color) The color assigned to gains (defaults to NULL)
`normal.color`	(color) The color assigned to normal ploidy(defaults to NULL)
`loss.color`	(colors) The color assigned to losses(defaults to NULL)
`r0`	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)(defaults to 0)
`r1`	(numeric) (karyoploteR parameter) r0 and r1 define the vertical range of the data panel to be used to draw this plot. They can be used to split the data panel in different vertical ranges (similar to tracks in a genome browser) to plot differents data. If NULL, they are set to the min and max of the data panel, it is, to use all the available space. (defaults to NULL)(defaults to 1)
`...`	The ellipsis operator can be used to specify any additional graphical parameters. Any additional parameter will be passed to the internal calls to karyoploteR functions.

Details

This function will open the HDF5 file, extract the CNV values and counts, for each genomic region, the number of cells with a gain or a loss. It then plots a histogram-like summary of these numbers.

Value

Invisibly returns the karyoplot object representing the plot. With it it is possible to add other elements to the plot using standrad karyoploteR functions. The returned object will have an additional "latest.plot" element with a list containing: the number of cells, the bin size used to partition the genome, the windows representing such partitioning with the number of cells with gains and losses in that each window, the number of bins per chromosome and a GRanges with the regions where no cell had any data, the no-call regions.

Note

If the file is open by any other application the function will fail.

Examples

```
kp <- plotKaryotype(plot.type=4, genome="hg38")
#NOT RUN - Using 10X example data from https://www.10xgenomics.com/resources/datasets/
#plotSingleCellCopyNumberSummary(kp, "breast_tissue_D_2k_cnv_data.h5")
```

`prepareLabels` *prepareLabels*

Description

Prepare the Labels to plot.

Usage

```
prepareLabels(labels, x)
```

Arguments

`labels`	(character) labels to plot. (defaults to NULL)
`x`	(list or GRangesList) The list or GRangesList where extracting the labels

Value

prepareLabels return a vector with the same length as the length of x. If labels is NULL and x have names, it will return the names of x. If there are not names it will return numbers as labels. If labes is not NULL, prepareLabels will cut or recycle the names as needed.

Examples

```
#List of GRanges
seg.data <- list(a = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,801
             b=regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,8014630,
prepareLabels(x = seg.data)

#GRangesList
seg.data <- GRangesList(a = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50
                b = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,8014

prepareLabels(x = seg.data)
```

`readHDF5Ztree` *readHDF5Ztree*

Description

Read the Ztree (scipy matrix representarion of a hierarchical clustering) contained in a 10X single-cell CNV HDF5 file and return a valid R hclust object representing the same tree.

More info on the Ztree matrix: https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-den

Usage

```
readHDF5Ztree(hdf5.file)
```

Arguments

`hdf5.file`	(H5IdComponent object) The result of opening an HDF5 file with rhdf5::H5Fopen(data.file). The file must contain a "tree" element with Ztree. For example, HDF5 files for CNV data produced by 10X CellRanger.

Value

A valid hclust object (standard R hierarchical clustering results)

Examples

```
not.run <- TRUE #We have no valid hdf5 file small enough to fit in a package
#Not Run
#data.file <- "path to a single-cell CNV data file"
#open.file <- rhdf5::H5Fopen(data.file)
#hc.tree <- readHDF5Ztree(open.file)
```

removeNAs *removeNAs*

Description

Removing NA from snp data.

Usage

```
removeNAs(snp.data, lrr.na = TRUE, baf.na = TRUE, id.na = TRUE, verbose = TRUE)
```

Arguments

snp.data	(GRanges, GRangesList or list) A GRanges, GRangesList or a list of GRanges.
lrr.na	(logical) Whether to remove the rows that have NAs in the lrr column. (defaults to TRUE)
baf.na	(logical) Whether to remove the rows that have NAs in the baf column. (defaults to TRUE)
id.na	(logical) Whether to remove the rows that have NAs in the id column. (defaults to TRUE)
verbose	(logical) Whether to show information messages. (defaults to TRUE)

Details

This function will remove rows with NA values in snp data. We can decide which columns we take into account to detect NAs, using the lrr.na, baf.na and id.na parameters.

Value

A GRanges or a list of GRanges where the NA values have been removed.

Examples

```
#GRanges
seg.data <- regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,8014630,1452
seg.data <- removeNAs(snp.data = seg.data)

#List of GRanges
seg.data <- list(a = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,801
            b=regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50000,8014630,
seg.data <- removeNAs(snp.data = seg.data)

#GRangesList
seg.data <- GRangesList(a = regioneR::toGRanges(data.frame(chr = c("chr1", "chr1", "chr2", "chr5"), start = c(0,50
seg.data <- removeNAs(snp.data = seg.data)
```

transformChr *transformChr*

Description

Transformation of the chromosomes

Usage

```
transformChr(chr, chr.transformation = "23:X,24:Y,25:MT")
```

Arguments

chr	(character) The chromosome names to transform
chr.transformation	(character) The transformation of the chromosomes names in a comma separated "key:value" format.(defaults to "23:X,24:Y,25:MT")

Details

This function will transform the name of the chormosomes, according to the parameter chr.transformation. This is useful in situations were files, such as .seg files as the ones generated by CNVkit, have the chromosome names as numbers. In order to transform the sexual and mitocondrial chromosomes from numbers back to characters, we can use this function. The exact transformation might be provided as described at https://cnvkit.readthedocs.io/en/stable/importexport.html#import-seg, (e.g. "key:value")

Value

A character vector where the name of the chromosomes have been changed according to chr.transformation parameter.

Examples

```
seg.file <- system.file("extdata", "DNACopy_output.seg", package = "CopyNumberPlots", mustWork = TRUE)
seg.data <- read.table(file = seg.file, sep = "\t", skip = 1, stringsAsFactors = FALSE)
colnames(seg.data) <-  c("ID", "chrom", "loc.start", "loc.end", "num.mark", "seg.mean")

# here we have the name of the chromosomes in a number format and how many segments are in each one.
table(seg.data$chrom)

# by this way we have the name 23 and 24 respectively transformed to X and Y.
seg.data$chrom <- transformChr(chr = seg.data$chrom, chr.transformation ="23:X,24:Y,25:MT")
table(seg.data$chrom)
```

UCSCStyle *UCSCStyle*

Description

Set the style of the chromosome names to UCSC ("chr1" instead of "1")

Usage

```
UCSCStyle(x)
```

Arguments

x	(GRanges, GRangesList or list of GRanges) The object to transform to UCSC style

Value

The same x object with the styles of the seqlevels set to UCSC

Examples

```
#GRanges
seg.data <- regioneR::toGRanges(data.frame(chr = c("1", "1", "2", "5"), start = c(0,50000,8014630,14523572), end =
seg.data <- UCSCStyle(seg.data)

#List of GRanges
seg.data <- list(a = regioneR::toGRanges(data.frame(chr = c("1", "1", "2", "5"), start = c(0,50000,8014630,1452357
             b=regioneR::toGRanges(data.frame(chr = c("1", "1", "2", "5"), start = c(0,50000,8014630,14523572), e
seg.data <- UCSCStyle(seg.data)

#GRangesList
seg.data <- GRangesList(a = regioneR::toGRanges(data.frame(chr = c("1", "1", "2", "5"), start = c(0,50000,8014630,
                b = regioneR::toGRanges(data.frame(chr = c("1", "1", "2", "5"), start = c(0,50000,8014630,1452357
seg.data <- UCSCStyle(seg.data)
```

Ztree2Hclust *Ztree2Hclust*

Description

Transform a Z matrix representing the hierarchical clustering of elements into a valid hclust R object. The Z representation is used by scipy hierarchical clustering and in 10X single-cell CNV HDF5 files.

More info on the Ztree matrix: https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-den

Usage

```
Ztree2Hclust(Ztree)
```

Arguments

Ztree	(matrix) A matrix with 4 rows and num.elements - 1 columns. The first and second row represent the elements or clusters merged at each step and the third row the distance between the merged elements. This is the format used by scipy hierarchical clustering.

Value

A valid hclust object (standard R hierarchical clustering results)

Examples

```
ztree <- matrix(c(0,4,0.1,2,
                  1,3,0.2,2,
                  5,2,0.3,2,
                  6,7,0.4,2),
                  nrow=4)

 hc.tree <- Ztree2Hclust(ztree)
 plot(hc.tree)
```

Index

colByCopyNumber, 3
computeHclustPlotOrder, 4

EnsemblStyle, 5

getBAFColumn, 5
getChrColumn, 6
getColumn, 7
getCopyNumberColors, *3*, 9, *41*
getCopyNumberColumn, 10
getEndColumn, 11
getIDColumn, 12
getLOHColumn, 13
getLRRColumn, 14
getPosColumn, 15
getSegmentValueColumn, 16
getStartColumn, 17

loadCopyNumberCalls, 18
loadCopyNumberCallsCnmops, 19
loadCopyNumberCallsCNVkit, 20
loadCopyNumberCallsDECoN, 21
loadCopyNumberCallsDNAcopy, 22
loadCopyNumberCallsPanelcnmops, 24
loadCopyNumberCallspennCNV, 25
loadCopyNumberCallsSeg, 26
loadSNPData, 27, *31*
loadSNPDataDNAcopy, 28
loadSNPDataFromVCF, 29, *31*

plotBAF, *3*, 30
plotCopyNumberCalls, 32, *41*
plotCopyNumberCallsAsLines, 34
plotCopyNumberSummary, 36
plotKaryotype, *30, 31*, *35, 36*, *39, 40*
plotLRR, 38
plotSingleCellCopyNumberCalls, 40
plotSingleCellCopyNumberSummary, 42
prepareLabels, 43

readHDF5Ztree, 44
removeNAs, 45

toGRanges, *3*, *31*
transformChr, 46

UCSCStyle, 47

Ztree2Hclust, 47