Package ‘treediff’

February 17, 2024

Title Testing Differences Between Families of Trees

Version 0.2.1

Date 2024-02-11

Description Perform test to detect differences in structure between families of trees. The method is based on cophenetic distances and aggregated Student's tests.

Depends R (>= 4.0.0)

Imports adjclust, BiocGenerics, csaw, data.table, dplyr, InteractionSet, limma, methods, stats, SummarizedExperiment, reshape2, testthat, rlang, HiCDOC, purrr

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

URL https://forgemia.inra.fr/scales/treediff

BugReports https://forgemia.inra.fr/scales/treediff/-/issues

NeedsCompilation no

Author Nathalie Vialaneix [aut, cre] (<https://orcid.org/0000-0003-1156-0639>),
Gwendaelle Cardenas [aut],
Marie Chavent [aut],
Sylvain Foissac [aut],
Pierre Neuvial [aut] (<https://orcid.org/0000-0003-3584-9998>),
Nathanael Randriamihamison [aut]

Maintainer Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

Repository CRAN

Date/Publication 2024-02-17 13:20:02 UTC

R topics documented:

`clusterTree` *Create hierarchical clustering trees for each cluster in a given matrix*

Description

This function creates a hierarchical clustering tree for each group in a given matrix. The function breaks the chromosome into clusters using the "broken stick" method and then converts the clusters into trees.

Usage

```
clusterTree(mat)
```

Arguments

`mat`	A matrix containing the data to cluster. It should have columns named ’index1’, ’index2’, ’chromosome and one column for each matrices.

Value

A list containing the following objects:

trees A list of hierarchical clustering trees, one for each cluster in the matrix.

metadata A data frame containing the following columns: names (name of each tree), chromosome, cluster, and file.

Examples

```
n <- 5

matrice <- matrix(runif(n*n), nrow = n, ncol = n)
matrice[lower.tri(matrice)] <- t(matrice)[lower.tri(matrice)]
matrice <- as.data.frame(matrice)
names(matrice) <- c("mat_1", "mat_2", "mat_3", "mat_4", "mat_5")

chromosome <- rep(1, n)
index1 <- sample(1:100, n, replace = TRUE)
index2 <- sample(1:100, n, replace = TRUE)

mat <- cbind(chromosome, index1, index2, matrice)
```

```
res <- clusterTree(mat)
```

HiC2Tree *Convert Hi-C to trees*

Description

This function converts Hi-C data into trees, using adjClust. It takes as input a file path vector, the format of the input data, the bin size of the Hi-C array, the chromosomes to be included in the analysis, and the number of replicates. It returns a list containing all trees, metadata, index and treediff results.

Usage

```
HiC2Tree(files, format, binsize = NULL, index = NULL, chromosomes, replicates)
```

Arguments

`files`	A character vector containing the file paths of the input data.
`format`	A character vector indicating the format of the input data: "tabular", "cooler", "juicer", or "HiC-Pro".
`binsize`	An integer indicating the bin size of the Hi-C matrix.
`index`	A character indicating the path of the index for the input data. Required (and used) only with the "HiC-Pro" format.
`chromosomes`	A vector containing the chromosomes to be included in the analysis.
`replicates`	An integer indicating the number of replicates to be used in `treediff`.

Value

A list containing:

`trees`	A list of all trees.
`metadata`	A data frame containing the following columns: names (name of each tree), chromosome, cluster, and file.
`index`	A data table containing the correspondence of each bin in the genome.
`testRes`	A list of treediff results for each cluster.

References

Christophe Ambroise, Alia Dehman, Pierre Neuvial, Guillem Rigaill, and Nathalie Vialaneix (2019) Adjacency-constrained hierarchical clustering of a band similarity matrix with application to genomics. *Algorithms for Molecular Biology*, **14**(22), 363–389.

Examples

```
replicates <- 1:3
cond <- c("90", "110")
all_begins <- interaction(expand.grid(replicates, cond), sep = "-")
all_begins <- as.character(all_begins)

# single chromosome
nb_chr <- 1
chromosomes <- 1:nb_chr
all_mat_chr <- lapply(chromosomes, function(chr) {
  all_mat <- lapply(all_begins, function(ab) {
    mat_file <- paste0("Rep", ab, "-chr", chr, "_200000.bed")
  })
  all_mat <- unlist(all_mat)
})
index <- system.file("extdata", "index.200000.longest18chr.abs.bed",
                     package = "treediff")
format <- rep("HiC-Pro", length(replicates) * length(cond) * nb_chr)
binsize <- 200000
files <- system.file("extdata", unlist(all_mat_chr), package = "treediff")
replicates <- c(3, 3)
HiC2Tree(files, format, binsize, index, chromosomes, replicates)

## Not run:
# two chromosomes
nb_chr <- 2
chromosomes <- 1:nb_chr
all_mat_chr <- lapply(chromosomes, function(chr) {
  all_mat <- lapply(all_begins, function(ab) {
    mat_file <- paste0("Rep", ab, "-chr", chr, "_200000.bed")
  })
  all_mat <- unlist(all_mat)
})
files <- system.file("extdata", unlist(all_mat_chr), package = "treediff")
format <- rep("HiC-Pro", length(replicates) * length(cond) * nb_chr)
replicates <- c(3, 3)
HiC2Tree(files, format, binsize, index, chromosomes, replicates)

## End(Not run)
```

`HiCDOCDataSet` *Create a HiCDOCDataSet object from a set of files*

Description

This function creates a count matrix from a set of files in different formats, such as tabular, cooler, juicer or HiC-Pro. It returns a list of interaction matrices.

Usage

```
HiCDOCDataSet(files, format, binsize = NULL, chromosomes, index = NULL)
```

Arguments

`files`	A character vector of file paths.
`format`	A character vector of file formats corresponding to the files in `file`. Supported formats are "tabular", "cooler", "juicer", and "HiC-Pro".
`binsize`	An integer representing the bin size to use for cooler and juicer formats. Ignored for tabular and HiC-Pro formats.
`chromosomes`	A character vector specifying the chromosomes to include in the output.
`index`	A character vector of file paths to the index files required for HiC-Pro format. Ignored for other formats.

Value

A list containing the following objects:

HiCDOCDataSet A list of interaction matrices of the HiCDOCDataSet class of the HiCDOC package, one for each file

indexData A data frame of index data for each interaction in the matrices.

index_mat_chr A data frame containing the name of the matrices and the corresponding chromosome.

Examples

```
## Not run:
replicates <- 1:2
cond <- "90"
all_begins <- interaction(expand.grid(replicates, cond), sep = "-")
all_begins <- as.character(all_begins)

nb_chr <- 2
chromosomes <- 1:nb_chr
all_mat_chr <- lapply(chromosomes, function(chr) {
  all_mat <- lapply(all_begins, function(ab) {
    mat_file <- paste0("Rep", ab, "-chr", chr, "_200000.bed")
  })
  all_mat <- unlist(all_mat)
})
index <- system.file("extdata", "index.200000.longest18chr.abs.bed",
                     package = "treediff")
format <- rep("HiC-Pro", length(replicates) * length(cond) * nb_chr)
binsize <- 200000
files <- system.file("extdata", unlist(all_mat_chr), package = "treediff")
HiCDOCDataSet(files, format, binsize, chromosomes, index)

## End(Not run)
```

`normalizeCount` *Normalize count matrix using cyclic loess*

Description

This function normalizes the count matrix using loess regression.

Usage

```
normalizeCount(count_matrice)
```

Arguments

`count_matrice`	The count matrix to normalize.

Value

count_matrice A data.frame of the normalized count matrix.

Examples

```
nb_row <- 120
chromosome <- rep(1, nb_row)
index1 <- sample(1:100, nb_row, replace = TRUE)
index2 <- sample(1:100, nb_row, replace = TRUE)

m <- data.frame("mat_1" =  sample(1:500, nb_row, replace = TRUE),
                "mat_2" =  sample(1:500, nb_row, replace = TRUE),
                "mat_3" =  sample(1:500, nb_row, replace = TRUE),
                "mat_4" =  sample(1:500, nb_row, replace = TRUE))

mat <- cbind(chromosome, index1, index2, m)

normalizeCount(mat)
```

`treediff` *Perform the treediff test*

Description

Perform the treediff test to compare two sets of trees.

Usage

```
treediff(trees1, trees2, replicates, scale = FALSE, order_labels = FALSE)

## S3 method for class 'treeTest'
print(x, ...)

## S3 method for class 'treeTest'
summary(object, ...)
```

Arguments

`trees1` A list of trees corresponding to the first condition (set). Trees are structured into groups (or clusters) with the same number of replicates in each group. Trees are ordered by groups and then by replicates: {group1+rep1, group1+rep2, ...}. One test is performed for each group.

`trees2` A list of trees corresponding to the second condition. Trees are also structured in groups (or clusters) that are exactly the same than for the first condition. The number of replicates in each group can be different from that of `trees1`.

`replicates` A numeric vector of length 2 with the number of replicates for each condition.

`scale` Logical. If `TRUE`, the trees are all rescaled to have a minimum height equal to 0 and a maximum height equal to 1. Default to `FALSE`.

`order_labels` Logical. If `TRUE`, align leaves ordering in all trees (required if your trees don't have their leaves ordered identically). Default to `FALSE`.

`x` a `treeTest` object to print

`...` not used

`object` a `treeTest` object to print

Details

This function compares two sets of trees using a p-value aggregation method. The p-values are obtained by the treediff method, as described in (Neuvial *et al.*, 2023).

Value

An object of class `treeTest` with the following entries:

`p.value` the p-value for the treediff test.

`statistic` the value of the Student's statistic of each leaf pair of the tree test.

`p.value.indiv` the p-value of the Student's test for each leaf pair of the tree test.

`method` a character string indicating what type of test was performed.

`data.name` a character string giving the names of the tree conditions.

Author(s)

Gwendaëlle Cardenas
Marie Chavent <marie.chavent@u-bordeaux.fr>
Sylvain Foissac <sylvain.foissac@inrae.fr>
Pierre Neuvial <pierre.neuvial@math.univ-toulouse.fr>
Nathanaël Randriamihamison
Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

References

Neuvial Pierre, Randriamihamison Nathanaël, Chavent Marie, Foissac Sylvain and Vialaneix Nathalie (2024) A two-sample tree-based test for hierarchically organized genomic signals. *Journal of the Royal Statistical Society, Series C*, *Forthcoming*.

Examples

```
leaves <- c(100, 120, 50, 80)

trees <- sapply(leaves, FUN = function(leaf) {
  base_data <- matrix(rnorm(2000), nrow = leaf, ncol = 200)

## generates two sets of trees with 4 clusters with 100, 120, 50 and 80
## leaves respectively
## 4 replicates in the first condition and 6 in the second condition

  set1 <- replicate(4, sample(1:100, 50, replace = FALSE))
  set2 <- replicate(6, sample(101:200, 50, replace = FALSE))

  trees1 <- apply(set1, 2, function(asample) {
    samples <- base_data[, asample]
    out <- hclust(dist(samples), method = "ward.D2")
    return(out)
  })

  trees2 <- apply(set2, 2, function(asample) {
    samples <- base_data[, asample]
    out <- hclust(dist(samples), method = "ward.D2")
    return(out)
  })
  return(list("trees1" = trees1, "trees2" = trees2))
})

trees1 <- unlist(trees[1, ], recursive = FALSE)
trees2 <- unlist(trees[2, ], recursive = FALSE)
replicates <- c(4, 6)

tree_pvals <- treediff(trees1, trees2, replicates)
## 4 p-values, one for each cluster
tree_pvals$p.value
```

Index