Package: clustur (via r-universe)

February 14, 2025

Type Package

Title Clustering

Version 0.1.2

Date 2024-11-25

Maintainer Patrick Schloss <`pschloss@umich.edu`>

Description A tool that implements the clustering algorithms from 'mothur' (Schloss PD et al. (2009) <doi:10.1128/AEM.01541-09>). 'clustur' make use of the cluster() and make.shared() command from 'mothur'. Our cluster() function has five different algorithms implemented: 'OptiClust', 'furthest', 'nearest', 'average', and 'weighted'. 'OptiClust' is an optimized clustering method for Operational Taxonomic Units, and you can learn more here, (Westcott SL, Schloss PD (2017) <doi:10.1128/mspheredirect.00073-17>). The make.shared() command is always applied at the end of the clustering command. This functionality allows us to generate and create clustering and abundance data efficiently.

License GPL (>= 3)

Encoding UTF-8

Imports Matrix, methods, Rcpp (>= 1.0.12), utils

Depends R (>= 3.5.0)

LinkingTo Rcpp, testthat

Suggests knitr, rmarkdown, testthat (>= 3.0.0), xml2

Config/testthat/edition 3

RoxygenNote 7.3.2

VignetteBuilder knitr

URL `http://www.schlosslab.org/clustur/`, `https://github.com/SchlossLab/clustur`

BugReports `https://github.com/SchlossLab/clustur/issues`

Config/pak/sysreqs make

Repository https://schlosslab.r-universe.dev

RemoteUrl https://github.com/schlosslab/clustur

RemoteRef HEAD

RemoteSha c65a071ccf6211a3bf324a233bee8df780b130a2

Contents

cluster 2
create_sparse_matrix 3
example_path 4
get_abundance 5
get_bins 5
get_count_table 6
get_cutoff 7
get_distance_df 7
get_metrics 8
read_count 9
read_dist 9
split_clusters_to_list 10
validate_count_table 11

Index **12**

`cluster` *Cluster entities together*

Description

Clusters entities represented in a distance matrix and count table using one of several algorithms and outputs information about the composition and abundance of each cluster

Usage

```
cluster(
  distance_object,
  cutoff,
  method = "opticlust",
  feature_column_name_to = "feature",
  bin_column_name_to = "bin",
  random_seed = 123
)
```

Arguments

`distance_object`	The distance object that was created using the ‘read_dist()‘ function.
`cutoff`	The cutoff you want to cluster towards.
`method`	The method of clustering to be performed: opticlust (default), furthest, nearest, average, or weighted.
`feature_column_name_to`	Set the name of the column in the cluster dataframe that contains the sequence names.
`bin_column_name_to`	Set the name of the column in the cluster dataframe that contains the name of the group of sequence names.
`random_seed`	the random seed to use, (default = 123).

Value

A list of ‘data.frames‘ that contain abundance, and clustering results. If you used ‘method = opticlust‘, it will also return clustering performance metrics.

Examples

```
cutoff <- 0.03
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff)

cluster_results <- cluster(distance_data,
                           cutoff, method = "opticlust",
                           feature_column_name_to = "sequence",
                           bin_column_name_to = "omu")
cluster_results <- cluster(distance_data,
                           cutoff, method = "furthest")
cluster_results <- cluster(distance_data,
                           cutoff, method = "nearest")
cluster_results <- cluster(distance_data,
                           cutoff, method = "average")
cluster_results <- cluster(distance_data,
                           cutoff, method = "weighted")
```

`create_sparse_matrix` *Create Sparse Matrix*

Description

Given a list of i indexes, j indexes, and distances values, we can create a sparse distance matrix for you. Each vector must have the same size.

Usage

```
create_sparse_matrix(i_index, j_index, distances)
```

Arguments

`i_index`	A list of i indexes, must be numeric
`j_index`	A list of j indexes, must be numeric
`distances`	A list of the distance at the i and j index

Value

a ‘dgTMatrix‘ from the ‘Matrix‘ library.

Examples

```
i_values <- as.integer(1:100)
j_values <- as.integer(sample(1:100, 100, TRUE))
x_values <- as.numeric(runif(100, 0, 1))
s_matrix <- create_sparse_matrix(i_values, j_values, x_values)
```

`example_path` *Example Path*

Description

This function was created as a helper function to generate file paths to our internal data. You should use this function if you want to follow along with the example, or interact with the data

Usage

```
example_path(file = NULL)
```

Arguments

`file`	The file name of the data; leave as NULL (default) to get full list of example files

Value

the path to the file as a ‘character‘ or a vector of ‘character‘ giving example filenames if ‘fill = NULL‘.

Examples

```
example_path("amazon_phylip.dist")
example_path()
```

`get_abundance` *Get Shared*

Description

GetShared returns the generated abundance ‘data.frame‘ from the ‘cluster()‘ function

Usage

```
get_abundance(cluster_data)
```

Arguments

`cluster_data`	The output from the ‘cluster()‘ function.

Value

a shared data.frame

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                          count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
shared <- get_abundance(df_clusters)
```

`get_bins` *Get Clusters*

Description

GetClusters returns a ‘data.frame‘ of the generated clusters from the ‘cluster()‘ function.

Usage

```
get_bins(cluster_data)
```

Arguments

`cluster_data`	The output from the ‘cluster()‘ function.

Value

the created cluster ‘data.frame‘.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
clusters <- get_bins(df_clusters)
```

`get_count_table` *Get Count Table*

Description

This function returns the count table that was used to generate the distance object.

Usage

```
get_count_table(distance_object)
```

Arguments

`distance_object`	The output from the ‘read.dist()‘ function.

Value

a count_table ‘data.frame‘.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
count_table <- get_count_table(distance_data)
```

`get_cutoff` *Get Cutoff*

Description

Returns the distance cutoff of the cluster object from the ‘cluster()‘ function

Usage

```
get_cutoff(cluster_data)
```

Arguments

`cluster_data`	The output from the ‘cluster()‘ function.

Value

the cutoff value as a ‘dbl‘

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
cutoff <- get_cutoff(df_clusters)
```

`get_distance_df` *Get Distance Data Frame*

Description

This function will generate a ‘data.frame‘ that contains the distances of all the indexes.

Usage

```
get_distance_df(distance_object)
```

Arguments

`distance_object`	The output from the ‘read.dist()‘ function.

Value

a distance ‘data.frame‘.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                                        count_table, cutoff, FALSE)
count_table <- get_count_table(distance_data)
```

`get_metrics` *Get Metrics*

Description

GetMetrics returns the generated metrics ‘data.frame‘ from the ‘cluster()‘ function.

Usage

```
get_metrics(cluster_data)
```

Arguments

`cluster_data`	The output from the ‘cluster()‘ function.

Value

a list of metric data.frames

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
df_clusters <- cluster(distance_data, cutoff, method = "opticlust")
list_of_metrics <- get_metrics(df_clusters)
```

`read_count` *Read count table*

Description

This function will read and return your count table. It can take in sparse and full count tables.

Usage

```
read_count(count_table_path)
```

Arguments

`count_table_path`	The file path of your count table.

Value

a count table ‘data.frame‘.

Examples

```
count_table <- read_count(example_path("amazon.full.count_table"))
```

`read_dist` *Read distance matrices*

Description

Read in distances from a file that is formatted with three columns for the row, column, and distance of a sparse, square matrix or in a phylip-formatted distance matrix.

Usage

```
read_dist(distance_file, count_table, cutoff, is_similarity_matrix = FALSE)
```

Arguments

`distance_file`	Either a phylip or column distance file, or a sparse matrix. The function will detect the format for you.
`count_table`	A table of names and the given abundance per group. Can be in mothur’s sparse or full format. The function will detect the format for you.
`cutoff`	The value you wish to use as a cutoff when clustering.
`is_similarity_matrix`	are you using a similarity matrix (default) or distance matrix?

Value

A distance ‘externalptr‘ object that contains all your distance information. Can be accessed using ‘get_distance_df()‘

Examples

```
i_values <- as.integer(1:100)
j_values <- as.integer(sample(1:100, 100, TRUE))
x_values <- as.numeric(runif(100, 0, 1))
s_matrix <- create_sparse_matrix(i_values, j_values, x_values)

sparse_count <- data.frame(
                Representative_Sequence = 1:100,
                total = rep(1, times = 100))

column_path <- example_path("amazon_column.dist")
phylip_path <- example_path("amazon_phylip.dist")
count_table <- read_count(example_path("amazon.full.count_table"))

data_column <- read_dist(column_path, count_table, 0.03)
data_phylip <- read_dist(phylip_path, count_table, 0.03)
data_sparse <- read_dist(s_matrix, sparse_count, 0.03)
```

`split_clusters_to_list` *Split Clusters to List*

Description

‘split_clusters_to_list()‘ will extract clusters from the cluster generated ‘data.frame‘. It will then turn those clusters into a list. This allows users to more easily visualize their data.

Usage

```
split_clusters_to_list(cluster)
```

Arguments

`cluster`	The output generated from the ‘cluster()‘ function.

Value

a named ‘list‘ of clusters.

Examples

```
cutoff <- 0.2
count_table <- read_count(example_path("amazon.full.count_table"))
distance_data <- read_dist(example_path("amazon_column.dist"),
                           count_table, cutoff, FALSE)
cluster_results <- cluster(distance_data, cutoff, method = "opticlust")

cluster_list <- split_clusters_to_list(cluster_results)
```

`validate_count_table` *Validate Count Table*

Description

If the count table is already valid nothing will change, otherwise it will add a new group to the count table file.

Usage

```
validate_count_table(count_table_df)
```

Arguments

`count_table_df`	The count table ‘data.frame‘ object.

Details

Determines whether user supplied count table is valid

Value

A validated count table ‘data.frame‘

Examples

```
count_table <- read.delim(example_path("amazon.full.count_table"))
count_table_valid <- validate_count_table(count_table)
```

Index

cluster, 2
create_sparse_matrix, 3

example_path, 4

get_abundance, 5
get_bins, 5
get_count_table, 6
get_cutoff, 7
get_distance_df, 7
get_metrics, 8

read_count, 9
read_dist, 9

split_clusters_to_list, 10

validate_count_table, 11