Package ‘CrossClustering’

May 14, 2024

**Type** Package

**Title** A Partial Clustering Algorithm

**Version** 4.1.2

**Date** 2024-05-01

**Maintainer** Paola Tellaroli <paola.tellaroli@gmail.com>

**Description** Provide the ‘CrossClustering’ algorithm (Tellaroli et al. (2016) <doi:10.1371/journal.pone.0152333>), which is a partial clustering algorithm that combines the Ward’s minimum variance and Complete Linkage algorithms, providing automatic estimation of a suitable number of clusters and identification of outlier elements.

**License** GPL-3

**URL** https://CRAN.R-project.org/package=CrossClustering

**BugReports** https://github.com/CorradoLanera/CrossClustering/issues

**Depends** R (>= 4.1)

**Imports** checkmate, cli, cluster, crayon, dplyr, flip, mclust, purrr, utils

**Suggests** covr, devtools, lintr, roxygen2, spelling, testthat, usethis

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**Language** en-US

**NeedsCompilation** no

**Author** Paola Tellaroli [cre, aut], Marco Buzzi [aut], Michele Donato [aut], Livio Finos [aut], Philippe Courcoux [aut], Corrado Lanera [aut]

**Repository** CRAN

**Date/Publication** 2024-05-14 09:30:19 UTC
\textbf{R topics documented:}

ari ................................................................. 2
cc_crossclustering .................................................. 4
cc_get_cluster ..................................................... 8
cc_test_ari .......................................................... 10
cc_test_ari_permutation .......................................... 11
chain_effect ....................................................... 12
consensus_cluster ............................................... 13
is_zero ............................................................. 14
nb_data ............................................................ 15
prune_zero_tail .................................................. 15
reverse_table ...................................................... 16
toy ................................................................. 17
twomoons .......................................................... 17
worms ............................................................... 18

\textbf{Index} 19

\begin{verbatim}
ari

\textit{ari} \hspace{1cm} Computes the adjusted Rand index and the confidence interval, comparing two classifications from a contingency table.

\textbf{Description}

Computes the adjusted Rand index and the confidence interval, comparing two classifications from a contingency table.

\textbf{print method for ari class}

\textbf{Usage}

\texttt{ari(mat, alpha = 0.05, digits = 2)}

\texttt{## S3 method for class 'ari'
print(x, ...)}

\textbf{Arguments}

\texttt{mat} \hspace{1cm} A matrix of integers representing the contingency table of reference

\texttt{alpha} \hspace{1cm} A single number strictly included between 0 and 1 representing the significance level of interest. (default is 0.05)

\texttt{digits} \hspace{1cm} An integer for the returned significant digits to return (default is 2)

\texttt{x} \hspace{1cm} an object used to select a method.

\texttt{...} \hspace{1cm} further arguments passed to or from other methods.
\end{verbatim}
**ari**

**Details**

The adjusted Rand Index (ARI) should be interpreted as follows:

- **ARI >= 0.90** excellent recovery
- **0.80 <= ARI < 0.90** good recovery
- **0.65 <= ARI < 0.80** moderate recovery
- **ARI < 0.65** poor recovery

As the confidence interval is based on the approximation to the Normal distribution, it is recommended to trust in the confidence interval only in cases of total number of object clustered greater than 100.

**Value**

An object of class `ari` with the following elements:

- **AdjustedRandIndex**
  - The adjusted Rand Index
- **CI**
  - The confidence interval

**Methods (by generic)**

- `print(ari):`

**Author(s)**

Paola Tellaroli, <paola dot tellaroli at unipd dot it>;

**References**

- D. Steinley, M.J. Brusco, L. Hubert (2016) The Variance of the Adjusted Rand Index, Psychological Methods, 21(2), 261-272

**Examples**

```r
### This example compares the adjusted Rand Index as computed on the
### partitions given by Ward's algorithm with the ground truth on the
### famous Iris data set by the adjustedRandIndex function
### mclust package and by the ari function.
library(CrossClustering)
library(mclust)

clusters <- iris[-5] |>
cc_crossclustering

A partial clustering algorithm with automatic estimation of the number of clusters and identification of outliers

Description

This function performs the CrossClustering algorithm. This method combines the Ward’s minimum variance and Complete-linkage (default, useful for finding spherical clusters) or Single-linkage (useful for finding elongated clusters) algorithms, providing automatic estimation of a suitable number of clusters and identification of outlier elements.

Usage

cc_crossclustering(
  dist,
  k_w_min = 2,
  k_w_max = attr(dist, "Size") - 2,
  k2_max = k_w_max + 1,
  out = TRUE,
  method = c("complete", "single")
)

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

Arguments

dist A dissimilarity structure as produced by the function dist
k_w_min (int) Minimum number of clusters for the Ward’s minimum variance method. By default is set equal 2
k_w_max (int) Maximum number of clusters for the Ward’s minimum variance method (see details)
k2_max  (int) Maximum number of clusters for the Complete/Single-linkage method. It can not be equal or greater than the number of elements to cluster (see details)
out  (lgI) If TRUE (default) outliers must be searched (see details)
method  (chr) "complete" (default) or "single". CrossClustering combines Ward’s algorithm with Complete-linkage if method is set to "complete", otherwise (if method is set to 'single') Single-linkage will be used.
x  an object used to select a method.
...  further arguments passed to or from other methods.

Details

See cited document for more details.

Value

A list of objects describing characteristics of the partitioning as follows:

Optimal_cluster  number of clusters
cluster_list_elements  a list of clusters; each element of this lists contains the indices of the elements belonging to the cluster
Silhouette  the average silhouette width over all the clusters
n_total  total number of input elements
n_clustered  number of input elements that have actually been clustered

Functions

• print(crossclustering):

Author(s)

Paola Tellaroli, <paola dot tellaroli at unipd dot it>; Marco Bazzi, <bazzi at stat dot unipd dot it>; Michele Donato, <mdonato at stanford dot edu>

References


Examples

```r
library(CrossClustering)

### Example of Cross-Clustering as in reference paper
### method = "complete"

data(toy)

### toy is transposed as we want to cluster samples (columns of the
### original matrix)
toy_dist <- t(toy) |> 
dist(method = "euclidean")

### Run CrossClustering
cross_clustering(  
  toy_dist,  
  k_w_min = 2,  
  k_w_max = 5,  
  k2_max = 6,  
  out = TRUE
)

### Simulated data as in reference paper
### method = "complete"
set.seed(10)
sg <- c(500, 250, 700, 300, 100)
# 5 clusters
t <- matrix(0, nrow = 5, ncol = 5)
t[1, ] <- rep(6, 5)
t[2, ] <- c(0, 5, 12, 13, 15)
t[3, ] <- c(15, 11, 9, 5, 0)
t[4, ] <- c(6, 12, 15, 10, 5)
t[5, ] <- c(12, 17, 3, 7, 10)

t_mat <- NULL
for (i in seq_len(nrow(t))) {
  t_mat <- rbind(    
    t_mat,    
    matrix(rep(t[i, ], sg[i]), nrow = sg[i], byrow = TRUE)
  )
}
data_15 <- matrix(NA, nrow = 2000, ncol = 5)
data_15[1:1850, ] <- matrix(    
  abs(rnorm(sum(sg) * 5, sd = 1.5)),    
  nrow = sum(sg),    
  ncol = 5
) + t_mat

set.seed(100) # simulate outliers
```
data_15[1851:2000,] <- matrix(
  runif(n = 150 * 5, min = 0, max = max(data_15, na.rm = TRUE)),
  nrow = 150,
  ncol = 5
)

### Run CrossClustering
cc_crossclustering(
  dist(data_15),
  k_w_min = 2,
  k_w_max = 19,
  k2_max = 20,
  out = TRUE
)

#### Correlation-based distance is often used in gene expression time-series data analysis. Here there is an example, using the "complete" method.

data(nb_data)
nb_dist <- as.dist(1 - abs(cor(t(nb_data))))
cc_crossclustering(dist = nb_dist, k_w_max = 20, k2_max = 19)

#### method = "single"
### Example on a famous shape data set
### Two moons data

data(twomoons)
moons_dist <- twomoons[, 1:2] |> dist(method = "euclidean")
cc_moons <- cc_crossclustering(
  moons_dist,
  k_w_max = 9,
  k2_max = 10,
  method = 'single'
)

moons_col <- cc_get_cluster(cc_moons)
plot(
  twomoons[, 1:2],
  col = moons_col,
  pch = 19,
  xlab = "",
  ylab = "",
  main = "CrossClustering-Single"
)

### Worms data
```r
data(worms)

worms_dist <- worms[, 1:2] |> dist(method = "euclidean")

c_worms <- cc_crossclustering(
  worms_dist,
  k_w_max = 9,
  k2_max = 10,
  method = "single"
)

worms_col <- cc_get_cluster(cc_worms)

plot(
  worms[, 1:2],
  col = worms_col,
  pch = 19,
  xlab = "",
  ylab = "",
  main = "CrossClustering-Single"
)

### CrossClustering-Single is not affected to chain-effect problem

data(chain_effect)

chain_dist <- chain_effect |> dist(method = "euclidean")

c_c_chain <- cc_crossclustering(
  chain_dist,
  k_w_max = 9,
  k2_max = 10,
  method = "single"
)

chain_col <- cc_get_cluster(cc_chain)

plot(
  chain_effect,
  col = chain_col,
  pch = 19,
  xlab = "",
  ylab = "",
  main = "CrossClustering-Single"
)
```

---

**cc_get_cluster**

Provides the vector of clusters' ID to which each element belong to.
**Description**

Provides the vector of clusters’ ID to which each element belong to.

**Usage**

```r
cc_get_cluster(x, n_elem)
```

```r
## Default S3 method:
cc_get_cluster(x, n_elem)
```

```r
## S3 method for class 'crossclustering'
cc_get_cluster(x, n_elem)
```

**Arguments**

- `x` list of clustered elements or a `crossclustering` object
- `n_elem` total number of elements clustered (ignored if `x` is of class `crossclustering`)

**Value**

An integer vector of clusters to which the elements belong (1 for the outliers, ID + 1 for the others).

**Methods (by class)**

- `cc_get_cluster(default)`: default method for `cc_get_cluster`.
- `cc_get_cluster(crossclustering)`: automatically extract inputs from a `crossclustering` object

**Author(s)**

Paola Tellaroli, <paola.tellaroli@unipd.it>; Marco Bazzi, <bazzi@stat.unipd.it>; Michele Donato, <mdonato@stanford.edu>.

**References**


**Examples**

```r
library(CrossClustering)

data(toy)

### toy is transposed as we want to cluster samples (columns of the
### original matrix)
toy_dist <- t(toy) |> dist(method = "euclidean")
```
### Run CrossClustering

toyres <- cc_crossclustering(
  toy_dist,
  k_w_min = 2,
  k_w_max = 5,
  k2_max = 6,
  out = TRUE
)

### cc_get_cluster
cc_get_cluster(toyres[, 7])

### cc_get_cluster directly from a crossclustering object
cc_get_cluster(toyres)

---

cc_test_ari

A test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

### Description

A test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

### Usage

cc_test_ari(ground_truth, partition)

### Arguments

- **ground_truth** (int) A vector of the actual membership of elements in clusters
- **partition** The partition coming from a clustering algorithm

### Value

A list with six elements:

- **Rand** the Rand Index
- **ExpectedRand** expected value of Rand Index
- **AdjustedRand** Adjusted Rand Index
- **var_ari** variance of Rand Index
- **nari** nari
- **p-value** the p-value of the test
Author(s)

Paola Tellaroli, <paola dot tellaroli at unipd dot it>; Philippe Courcoux, <philippe dot courcoux at oniris-nantes dot fr>

References


Examples

library(CrossClustering)

clusters <- iris[-5] |> dist() |> hclust(method = 'ward.D') |> cutree(k = 3)

ground_truth <- iris[[5]] |> as.numeric()

cc_test_ari(ground_truth, clusters)

cc_test_ari_permutation

A permutation test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

Description

A permutation test for testing the null hypothesis of random agreement (i.e., adjusted Rand Index equal to 0) between two partitions.

Usage

cc_test_ari_permutation(ground_truth, partition)

Arguments

ground_truth (int) A vector of the actual membership of elements in clusters
partition The partition coming from a clustering algorithm
Value

A data_frame with two columns:

- **ari**: the adjusted Rand Index
- **p_value**: the p-value of the test

Author(s)

Paola Tellaroli, <paola dot tellaroli at unipd dot it>; Livio Finos, <livio dot finos at unipd dot it>

References


Examples

```r
library(CrossClustering)

clusters <- iris[-5] |> 
  dist() |> 
  hclust(method = 'ward.D') |> 
  cutree(k = 3)

ground_truth <- iris[[5]] |> 
  as.numeric()

cc_test_ari_permutation(ground_truth, clusters)
```

---

**chain_effect**

A toy dataset for illustrating the chain effect.

**Description**

A toy dataset for illustrating the chain effect.

**Usage**

`chain_effect`

**Format**

A data frame with 28 rows and 2 variables:

- **X**: numx coordinates 0 is negative.
- **Y**: numy coordinates.
consensus_cluster

Get clusters which reach max consensus

Description

Computes the consensus between Ward’s minimum variance and Complete-linkage (or Single-linkage) algorithms (i.e., the number of elements classified together by both algorithms).

Usage

consensus_cluster(k, cluster_ward, cluster_other)

Arguments

k (int) a vector containing the number of clusters for Ward and for Complete-linkage (or Single-linkage) algorithms, respectively
clustering_ward an object of class hclust for the Ward algorithm
clustering_other an object of class hclust for the Complete-linkage (or Single-linkage) algorithm

Value

an object of class consensus_cluster with the following elements:

elements list of the elements belonging to each cluster
; 
 a_star contingency table of the clustering 
; 
 max_consensus maximum clustering consensus
.

Author(s)

Paola Tellaroli, <paola dot tellaroli at unipd dot it>; Marco Bazzi, <bazzi at stat dot unipd dot it>; Michele Donato, <mdonato at stanford dot edu>.

References

is_zero

Check for zero

Check if a given, single, number is 0 or not

Usage

is_zero(num)

Arguments

num a numerical vector of length one

Value

a boolean, TRUE if num is 0

Examples

is_zero(1)

is_zero(0)
### nb_data

**RNA-Seq dataset example**

**Description**

nb_data contains a subset of a bigger normalized negative binomial simulated dataset.

**Usage**

nb_data

**Format**

A data frame with 100 observations on 36 numeric variables.

**Details**

This dataset is part of a larger simulated and normalized dataset with 2 experimental groups, 6 time-points and 3 replicates. Simulation has been done by using a negative binomial distribution. The first 20 genes are simulated with changes among time.

**Source**

Data included in the bioconductor package maSigPro. [https://doi.org/doi:10.18109/B9.bioc.maSigPro](https://doi.org/doi:10.18109/B9.bioc.maSigPro)

---

### prune_zero_tail

**Prune tail made of zeros**

**Description**

Given a diagonal matrix which is supposed to have no non-zero entry in the diagonal after the first one (if any) the function returns the diagonal (sub-)matrix without the columns and the row corresponding to the zero-entries in the diagonal (if any).

**Usage**

prune_zero_tail(diag_mat)

**Arguments**

- **diag_mat**
  
a diagonal matrix which must satisfy the following property: in the diagonal, every element after a zero is a zero.
Value

a diagonal matrix without zeros in the diagonal, composed by the first rows and columns of the original matrix with non zeros in the diagonal (which are also the only ones)

Examples

diag_mat <- diag(c(1, 2, 3, 0, 0, 0, 0))
prune_zero_tail(diag_mat)

reverse_table

Reverse the process of create a contingency table

Description

Reverse the process of create a contingency table

Usage

reverse_table(x)

Arguments

x a contingency table

Value

a list of 2 vector corresponding to the unrolled table

Examples

clust_1 <- iris[, 1:4] |> dist() |> hclust() |> cutree(k = 3)
clust_2 <- iris[, 1:4] |> dist() |> hclust() |> cutree(k = 4)
cont_table <- table(clust_1, clust_2)
reverse_table(cont_table)
toy

**A toy example matrix**

**Description**

A toy example matrix

**Usage**

toy

**Format**

A matrix of 10 row and 7 columns

twomoons

**A famous shape data set containing two clusters with two moons shapes and outliers**

**Description**

A famous shape data set containing two clusters with two moons shapes and outliers

**Usage**

twomoons

**Format**

A data frame with 52 rows and 3 variables:

- x  numx coordinates
- y  numy coordinates.
- clusters  integer cluster membership (outliers classified as 3rd cluster).
Description

A famous shape data set containing two clusters with two worms shapes and outliers

Usage

`worms`

Format

A data frame with 87 rows and 3 variables:

- `x` numx coordinates
- `y` numy coordinates.
- `cluster` integer cluster membership (outliers classified as 3rd cluster).
Index

* datasets
  chain_effect, 12
  nb_data, 15
  toy, 17
  twomoons, 17
  worms, 18

ari, 2

crossclustering, 4
cc_get_cluster, 8, 9
cc_test_ari, 10
cc_test_ari_permutation, 11
chain_effect, 12
consensus_cluster, 13

is_zero, 14

nb_data, 15

print.ari(ari), 2
print.crossclustering
  (cc_crossclustering), 4
prune_zero_tail, 15

reverse_table, 16

toy, 17

twomoons, 17

worms, 18