

Package ‘ClusterStability’

July 21, 2025

Type Package

Depends R ($\geq 2.2.4$), Rcpp, cluster, copula (≥ 0.999),
WeightedCluster

LinkingTo Rcpp

Title Assessment of Stability of Individual Objects or Clusters in
Partitioning Solutions

Version 1.0.4

Date 2023-03-07

Author Etienne Lord, Matthieu Willems, Francois-Joseph Lapointe, and Vladimir
Makarenkov

Maintainer Etienne Lord <m.etienne.lord@gmail.com>

Description Allows one to assess the stability of individual objects, clusters
and whole clustering solutions based on repeated runs of the K-means and K-medoids
partitioning algorithms.

License GPL-3

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-03-07 23:20:02 UTC

Contents

ClusterStability-package	2
calinski_harabasz_score	3
ClusterStability	3
ClusterStability_exact	4
davies_bouldin_score	5
dunn_score	6
Kcombination	6
Reorder	7
Stirling2nd	8
Undocumented functions	8

ClusterStability-package

Assessment of the stability of individual objects, clusters and a whole clustering solution based on repeated runs of a clustering algorithm.

Description

The ClusterStability package uses a probabilistic framework and some well-known clustering criteria (e.g. Calinski-Harabasz, Silhouette, Dunn and Davies-Bouldin) to compute the stability scores (*ST*) of each individual object (i.e., *element*) in the clustering solution provided by the K-means and K-medoids partitioning algorithms.

Details

Package:	ClusterStability
Type:	Package
Version:	1.0.2
Date:	2015-10-14
License:	GPL-2
Maintainer:	Etienne Lord <m.etienne.lord@gmail.com>, Vladimir Makarencov <makarencov.vladimir@uqam.ca>

Function [ClusterStability](#) computes the individual and global stability scores (*ST*) for a partitioning solution using either K-means or K-medoids (the approximate solution is provided).

Function [ClusterStability_exact](#) is similar to the [ClusterStability](#) function but uses the Stirling numbers of the second kind to compute the exact stability scores (but is limited to a small number of objects).

Function [Kcombination](#) computes the *k*-combination of a set of numbers for a given *k*.

Function [Reorder](#) returns the re-ordered partitioning of a series of clusters.

Function [Stirling2nd](#) computes the Stirling numbers of the second kind.

Author(s)

Etienne Lord, François-Joseph Lapointe and Vladimir Makarencov

See Also

[ClusterStability](#), [ClusterStability_exact](#), [Kcombination](#), [Reorder](#), [Stirling2nd](#)

calinski_harabasz_score

This function returns the Calinski Harabasz score.

Description

This function returns the Calinski Harabasz score of a partition (also known as the Variance Ratio Criterion).

Usage

```
calinski_harabasz_score(X, labels)
```

Arguments

X	the input dataset: either a matrix or a dataframe.
labels	the partition vector.

Value

The Calinski Harabasz score for this data.

References

T. Calinski and J. Harabasz. A dendrite method for cluster analysis. Communications in Statistics, 3, no. 1:1–27, 1974

Examples

```
calinski_harabasz_score(iris[1:10,1:4], c(3,2,2,2,3,1,2,3,2,2))
# Expected : 11.34223
```

ClusterStability

Calculates the approximate stability score (ST) of individual objects in a clustering solution (the approximate version allowing one to avoid possible variable overflow errors).

Description

This function will return the individual stability score *ST* and the global score *STglobal* using either the K-means or K-medoids algorithm and four different clustering indices: Calinski-Harabasz, Silhouette, Dunn or Davies-Bouldin.

Usage

```
ClusterStability(dat, k, replicate, type)
```

Arguments

<code>dat</code>	the input dataset: either a matrix or a dataframe.
<code>k</code>	the number of classes for the K-means or K-medoids algorithm (default=3).
<code>replicate</code>	the number of replicates to perform (default=1000).
<code>type</code>	the algorithm used in the partitioning: either 'kmeans' or 'kmedoids' algorithm (default=kmeans).

Value

Returns the individual (*ST*) and global (*ST_global*) stability scores for the four clustering indices: Calinski-Harabasz (*ch*), Silhouette (*sil*), Dunn (*dunn*) or Davies-Bouldin (*db*).

Examples

```
## Calculates the stability scores of individual objects of the Iris dataset
## using K-means, 100 replicates (random starts) and k=3
ClusterStability(dat=iris[1:4],k=3,replicate=100,type='kmeans');
```

ClusterStability_exact

Calculates the exact stability score (ST) for individual objects in a clustering solution.

Description

This function will return the exact individual stability score *ST* and the exact global score *ST_global* using either the K-means or K-medoids algorithm and four different clustering indices: Calinski-Harabasz, Silhouette, Dunn or Davies-Bouldin. **Variable overflow errors are possible for large numbers of objects.**

Usage

```
ClusterStability_exact(dat, k, replicate, type)
```

Arguments

<code>dat</code>	the input dataset: either a matrix or a dataframe.
<code>k</code>	the number of classes for the K-means or K-medoids algorithm (default=3).
<code>replicate</code>	the number of replicates to perform (default=1000).
<code>type</code>	the algorithm used in the partitioning: either 'kmeans' or 'kmedoids' algorithm (default=kmeans).

Value

Returns the exact individual (*ST*) and global (*ST_global*) stability scores for the four clustering indices: Calinski-Harabasz (*ch*), Silhouette (*sil*), Dunn (*dunn*) or Davies-Bouldin (*db*).

Examples

```
## Calculate the stability scores of individual objects of the Iris dataset
## using K-means, 100 replicates (random starts) and k=3
ClusterStability_exact(dat=iris[1:4],k=3,replicate=100,type='kmeans');
```

`davies_bouldin_score` *This function returns the Davies Bouldin score.*

Description

This function returns the Davies Bouldin score of a partition.

Usage

```
davies_bouldin_score(X, labels)
```

Arguments

<code>X</code>	the input dataset: either a matrix or a dataframe.
<code>labels</code>	the partition vector.

Value

The Davies Bouldin score for this data.

References

D. L. Davies and D. W. Bouldin. A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1, no. 2:224-227, 1979

Examples

```
davies_bouldin_score(iris[1:10,1:4], c(3,2,2,2,3,1,2,3,2,2))
# Expected : 0.5103277
```

dunn_score	<i>This function returns the Dunn_score.</i>
------------	--

Description

This function returns the Dunn score (also known as the e Dunn index) of a partition .

Usage

```
dunn_score(X, labels)
```

Arguments

- X the input dataset: either a matrix or a dataframe.
- labels the partition vector.

Value

The Dunn index score for this data.

References

J. Dunn. Well separated clusters and optimal fuzzy partitions. Journal of Cybernetics, 4:95–104, 1974.

Examples

```
dunn_score(iris[1:10,1:4], c(3,2,2,2,3,1,2,3,2,2))  
# Expected : 0.5956834
```

Kcombination	<i>Kcombination returns the list of all possible combinations of a set of numbers of a given length k.</i>
--------------	--

Description

This function, given a vector of numbers, will return all the possible combinations of a given length *k*.

Usage

```
Kcombination(data, k, selector)
```

Arguments

data the vector of numbers (*i.e.* elements) to consider.
 k the length of the returned combination (between 2 and 6 in this version).
 selector if set, returns only the combinations containing this number.

Value

Return a list of all possible combinations for the given vector of numbers.

Examples

```
## Returns the k-combination of the list of numbers: 1,2,3 of length=2.
## i.e. (1,2), (1,3), (2,3)
Kcombination(c(1,2,3),k=2)
## Returns only the k-combination containing the number 1.
## i.e. (1,2), (1,3)
Kcombination(c(1,2,3),k=2,selector=1)
```

Reorder	<i>This function returns the ordering of a partitioning solution in ascending order.</i>
---------	--

Description

This function returns the ordered partition of a set of numbers in ascending order and reordered to start at one. This is an auxiliary function.

Usage

```
Reorder(data)
```

Arguments

data vector of partition numbers to reorder.

Value

A vector of ordered partition numbers for this data.

Examples

```
Reorder(c(1,3,4,4,3,1))
# Expected : 1 2 3 3 2 1
```

Stirling2nd

Stirling2nd function computes the Stirling numbers of the second kind.

Description

This function returns the estimated Stirling numbers of the second kind *i.e.*, the number of ways of partitioning a set of n objects into k nonempty groups.

Usage

```
Stirling2nd(n,k)
```

Arguments

n	number of objects.
k	number of groups (<i>i.e.</i> classes).

Value

The Stirling number of the 2nd kind for n elements and k groups or *NaN* (if the Stirling number for those n and k is greater than 1e300).

Examples

```
Stirling2nd(n=3,k=2)
# Expected value=3
Stirling2nd(n=300,k=20)
# Expected value=NaN
```

Undocumented functions

Undocumented functions

Description

The following functions are for internal computation only: *calculate_global_PSG*, *calculate_indices*, *calculate_singleton*, *is_partition_group*, *p_n_k*, *p_tilde_n_k*, *calculate_individual_PSG_approximative*, *calculate_individual_PSG_exact*, *calculate_individual_PSG*.

Index

- * **k-combination**
 - Kcombination, [6](#)
- * **package**
 - ClusterStability-package, [2](#)
- * **partitioning criteria**
 - ClusterStability-package, [2](#)
- * **stability score**
 - ClusterStability-package, [2](#)

a2combination (Undocumented functions), [8](#)

calculate_global_PSG (Undocumented functions), [8](#)

calculate_indices (Undocumented functions), [8](#)

calculate_individual_PSG (Undocumented functions), [8](#)

calculate_individual_PSG_approximative (Undocumented functions), [8](#)

calculate_individual_PSG_exact (Undocumented functions), [8](#)

calculate_singleton (Undocumented functions), [8](#)

calinski_harabasz_score, [3](#)

ClusterStability, [2](#), [3](#)

ClusterStability-package, [2](#)

ClusterStability_exact, [2](#), [4](#)

davies_bouldin_score, [5](#)

dunn_score, [6](#)

is_partition_group (Undocumented functions), [8](#)

Kcombination, [2](#), [6](#)

p_n_k (Undocumented functions), [8](#)

p_tilde_n_k (Undocumented functions), [8](#)

Reorder, [2](#), [7](#)

Stirling2nd, [2](#), [8](#)

Undocumented functions, [8](#)