Package 'genieclust'

February 23, 2026

Type Package

Title Genie: Fast and Robust Hierarchical Clustering

Version 1.3.0

Date 2026-02-23

Description Genie is a robust hierarchical clustering algorithm (Gagolewski, Bartoszuk, Cena, 2016 <DOI:10.1016/j.ins.2016.05.003>). 'genieclust' is its faster, more capable implementation (Gagolewski, 2021 <DOI:10.1016/j.softx.2021.100722>). It enables clustering with respect to mutual reachability distances, allowing it to act as an alternative to 'HDBSCAN*' that can identify any number of clusters or their entire hierarchy. When combined with the 'deadwood' package, it can act as an outlier detector. Additional package features include the Gini and Bonferroni inequality indices, external cluster validity measures (e.g., the normalised clustering accuracy, the adjusted Rand index, the Fowlkes-Mallows index, and normalised mutual information), and internal cluster validity indices (e.g., the Calinski-Harabasz, Davies-Bouldin, Ball-Hall, Silhouette, and generalised Dunn indices). The 'Python' version of 'genieclust' is available via 'PyPI'.

BugReports https://github.com/gagolews/genieclust/issues

URL https://genieclust.gagolewski.com/, https://clustering-benchmarks.gagolewski.com/, https://github.com/gagolews/genieclust

License AGPL-3

Imports Rcpp, stats, utils, deadwood

Suggests datasets,

LinkingTo Rcpp

Encoding UTF-8

SystemRequirements OpenMP

RoxygenNote 7.3.3

NeedsCompilation yes

Author Marek Gagolewski [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-0637-6028>),
Maciej Bartoszuk [ctb],
Anna Cena [ctb],
Peter M. Larsen [ctb]

Maintainer Marek Gagolewski <marek@gagolewski.com>

Repository CRAN

Date/Publication 2026-02-23 17:00:20 UTC

Contents

`cluster_validity` *Internal Cluster Validity Measures*

Description

Implementation of cluster validity indices reviewed in (Gagolewski, Bartoszuk, Cena, 2021). See Section 2 therein for the respective definitions.

The greater the index value, the more *valid* (whatever that means) the assessed partition. For consistency, the Ball-Hall and Davies-Bouldin indexes as well as the within-cluster sum of squares (WCSS) take negative values.

Usage

```
calinski_harabasz_index(X, y)

dunnowa_index(
  X,
  y,
  M = 25L,
  owa_numerator = "SMin:5",
  owa_denominator = "Const"
)

generalised_dunn_index(X, y, lowercase_d, uppercase_d)

negated_ball_hall_index(X, y)

negated_davies_bouldin_index(X, y)
```

```
negated_wcss_index(X, y)

silhouette_index(X, y)

silhouette_w_index(X, y)

wcnn_index(X, y, M = 25L)
```

Arguments

`X`	numeric matrix with `n` rows and `d` columns, representing `n` points in a `d`-dimensional space
`y`	vector of `n` integer labels, representing a partition whose *quality* is to be assessed; `y[i]` is the cluster ID of the `i`-th point, `X[i, ]`; `1 <= y[i] <= K`, where `K` is the number or clusters
`M`	number of nearest neighbours
`owa_numerator, owa_denominator`	single string specifying the OWA operators to use in the definition of the DuNN index; one of: `"Mean"`, `"Min"`, `"Max"`, `"Const"`, `"SMin:D"`, `"SMax:D"`, where `D` is an integer defining the degree of smoothness
`lowercase_d`	an integer between 1 and 5, denoting d_1, ..., d_5 in the definition of the generalised Dunn (Bezdek-Pal) index (numerator: min, max, and mean pairwise intracluster distance, distance between cluster centroids, weighted point-centroid distance, respectively)
`uppercase_d`	an integer between 1 and 3, denoting D_1, ..., D_3 in the definition of the generalised Dunn (Bezdek-Pal) index (denominator: max and min pairwise intracluster distance, average point-centroid distance, respectively)

Value

A single numeric value (the more, the *better*).

Author(s)

Marek Gagolewski and other contributors

References

Ball, G.H., Hall, D.J., *ISODATA: A novel method of data analysis and pattern classification*, Technical report No. AD699616, Stanford Research Institute, 1965.

Bezdek, J., Pal, N., Some new indexes of cluster validity, *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 28, 1998, 301-315, doi:10.1109/3477.678624.

Calinski, T., Harabasz, J., A dendrite method for cluster analysis, *Communications in Statistics* 3(1), 1974, 1-27, doi:10.1080/03610927408827101.

Davies, D.L., Bouldin, D.W., A Cluster Separation Measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1 (2), 1979, 224-227, doi:10.1109/TPAMI.1979.4766909.

Dunn, J.C., A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, *Journal of Cybernetics* 3(3), 1973, 32-57, doi:10.1080/01969727308546046.

Gagolewski, M., Bartoszuk, M., Cena, A., Are cluster validity measures (in)valid?, *Information Sciences* 581, 620-636, 2021, doi:10.1016/j.ins.2021.10.004; preprint: https://raw.githubusercontent.com/gagolews/bibliography/master/preprints/2021cvi.pdf.

Gagolewski, M., A Framework for Benchmarking Clustering Algorithms, *SoftwareX* 20, 2022, 101270, doi:10.1016/j.softx.2022.101270, https://clustering-benchmarks.gagolewski.com.

Rousseeuw, P.J., Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis, *Computational and Applied Mathematics* 20, 1987, 53-65, doi:10.1016/03770427(87)901257.

See Also

The official online manual of **genieclust** at https://genieclust.gagolewski.com/

Gagolewski, M., **genieclust**: Fast and robust hierarchical clustering, *SoftwareX* 15:100722, 2021, doi:10.1016/j.softx.2021.100722

Examples

```
X <- as.matrix(iris[,1:4])
X[,] <- jitter(X)  # otherwise we get a non-unique solution
y <- as.integer(iris[[5]])
calinski_harabasz_index(X, y)  # good
calinski_harabasz_index(X, sample(1:3, nrow(X), replace=TRUE))  # bad
```

---

compare_partitions *External Cluster Validity Measures and Pairwise Partition Similarity Scores*

Description

The functions described in this section quantify the similarity between two label vectors `x` and `y` which represent two partitions of a set of n elements into, respectively, K and L nonempty and pairwise disjoint subsets; for a review, refer to the paper by Gagolewski (2025).

For instance, `x` and `y` can represent two clusterings of a dataset with n observations specified by two vectors of labels. The functions described here can be used as external cluster validity measures, where we assume that `x` is a reference (ground-truth) partition whilst `y` is the vector of predicted cluster memberships.

All indices except `normalized_clustering_accuracy()` can act as a pairwise partition similarity score: they are symmetric, i.e., `index(x, y) == index(y, x)`.

Each index except `mi_score()` (which computes the mutual information score) outputs 1 given two identical partitions. Note that partitions are always defined up to a permutation (bijection) of the set of possible labels, e.g., (1, 1, 2, 1) and (4, 4, 2, 4) represent the same 2-partition.

Usage

```
normalized_clustering_accuracy(x, y = NULL)

normalized_pivoted_accuracy(x, y = NULL)

pair_sets_index(x, y = NULL, simplified = FALSE, clipped = TRUE)

adjusted_rand_score(x, y = NULL, clipped = FALSE)

rand_score(x, y = NULL)

adjusted_fm_score(x, y = NULL, clipped = FALSE)

fm_score(x, y = NULL)

mi_score(x, y = NULL)

normalized_mi_score(x, y = NULL)

adjusted_mi_score(x, y = NULL, clipped = FALSE)

normalized_confusion_matrix(x, y = NULL)

normalizing_permutation(x, y = NULL)
```

Arguments

x	an integer vector of length n (or an object coercible to) representing a K-partition of an n-set (e.g., a reference partition), or a confusion matrix with K rows and L columns (see `table(x, y)`)
y	an integer vector of length n (or an object coercible to) representing an L-partition of the same set (e.g., the output of a clustering algorithm we wish to compare with x), or NULL (if x is an $K \times L$ confusion matrix)
simplified	whether to assume E=1 in the definition of the pair sets index index, i.e., use Eq. (20) in (Rezaei, Franti, 2016) instead of Eq. (18)
clipped	whether the result should be clipped to the unit interval, i.e., [0, 1]

Details

`normalized_clustering_accuracy()` is an asymmetric external cluster validity measure proposed by Gagolewski (2025). It assumes that the label vector x (or rows in the confusion matrix) represents the reference (ground truth) partition. It is the average proportion of correctly classified points in each cluster above the worst case scenario representing the uniform membership assignment, with the cluster ID matching based on the solution to the maximal linear sum assignment problem; see `normalized_confusion_matrix`). The index is given by: $\max_\sigma \frac{1}{K} \sum_{j=1}^{K} \frac{c_{\sigma(j),j} - c_{\sigma(j),\cdot}/K}{c_{\sigma(j),\cdot} - c_{\sigma(j),\cdot}/K}$, where C is a confusion matrix with K rows and columns, σ is a permutation of the set $\{1, \ldots, K\}$, and $c_{i,\cdot} = c_{i,1} + ... + c_{i,K}$ is the i-th row sum, under the assumption that $0/0 = 0$.

`normalized_pivoted_accuracy()` is defined as $\max_\sigma \sum_{j=1}^{K} \frac{c_{\sigma(j),j}/n - 1/K}{1-1/K}$, where σ is a permutation of the set $\{1, \ldots, K\}$, and n is the sum of all elements in C.

`pair_sets_index()` (PSI) was introduced by Rezaei and Franti (2016). The simplified PSI assumes E=1 in the definition of the index, i.e., uses Eq. (20) in the said paper instead of Eq. (18). For non-square matrices, missing rows/columns are assumed to be filled with 0s.

`rand_score()` gives the Rand score (the "probability" of agreement between the two partitions) and `adjusted_rand_score()` is its version corrected for chance (see Hubert, Arabie, 1985): its expected value is 0 given two independent partitions. Due to the adjustment, the resulting index may be negative for some inputs.

Similarly, `fm_score()` gives the Fowlkes-Mallows (FM) score and `adjusted_fm_score()` is its adjusted-for-chance version; (see Hubert, Arabie, 1985).

`mi_score()`, `adjusted_mi_score()` and `normalized_mi_score()` are information-theoretic scores, based on mutual information, see the definition of AMI_{sum} and NMI_{sum} in the paper by Vinh et al. (2010).

`normalized_confusion_matrix()` computes the confusion matrix and permutes its rows and columns so that the sum of the elements of the main diagonal is the largest possible (by solving the maximal assignment problem). The function only accepts $K \leq L$. The reordering of the columns of a confusion matrix can be determined by calling `normalizing_permutation()`.

Also note that the built-in `table()` function determines the standard confusion matrix.

Value

Each cluster validity measure is a single numeric value.

`normalized_confusion_matrix()` returns a numeric matrix.

`normalizing_permutation()` returns a vector of indexes.

Author(s)

Marek Gagolewski and other contributors

References

Gagolewski, M., A framework for benchmarking clustering algorithms, *SoftwareX* 20, 2022, 101270, doi:10.1016/j.softx.2022.101270, https://clustering-benchmarks.gagolewski.com.

Gagolewski, M., Normalised clustering accuracy: An asymmetric external cluster validity measure, *Journal of Classification* 42, 2025, 2-30. doi:10.1007/s00357024094822.

Hubert, L., Arabie, P., Comparing partitions, *Journal of Classification* 2(1), 1985, 193-218, esp. Eqs. (2) and (4).

Meila, M., Heckerman, D., An experimental comparison of model-based clustering methods, *Machine Learning* 42, 2001, pp. 9-29, doi:10.1023/A:1007648401407.

Rezaei, M., Franti, P., Set matching measures for external cluster validity, *IEEE Transactions on Knowledge and Data Mining* 28(8), 2016, 2173-2186.

Steinley, D., Properties of the Hubert-Arabie adjusted Rand index, *Psychological Methods* 9(3), 2004, pp. 386-396, doi:10.1037/1082989X.9.3.386.

Vinh, N.X., Epps, J., Bailey, J., Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, *Journal of Machine Learning Research* 11, 2010, 2837-2854.

See Also

The official online manual of **genieclust** at https://genieclust.gagolewski.com/

Gagolewski, M., **genieclust**: Fast and robust hierarchical clustering, *SoftwareX* 15:100722, 2021, doi:10.1016/j.softx.2021.100722

Examples

```
y_true <- iris[[5]]
y_pred <- kmeans(as.matrix(iris[1:4]), 3)$cluster
normalized_clustering_accuracy(y_true, y_pred)
normalized_pivoted_accuracy(y_true, y_pred)
pair_sets_index(y_true, y_pred)
pair_sets_index(y_true, y_pred, simplified=TRUE)
adjusted_rand_score(y_true, y_pred)
rand_score(table(y_true, y_pred)) # the same
adjusted_fm_score(y_true, y_pred)
fm_score(y_true, y_pred)
mi_score(y_true, y_pred)
normalized_mi_score(y_true, y_pred)
adjusted_mi_score(y_true, y_pred)
normalized_confusion_matrix(y_true, y_pred)
normalizing_permutation(y_true, y_pred)
```

---

gclust *Hierarchical Clustering Algorithm Genie*

Description

A reimplementation of *Genie* - a robust and outlier resistant clustering algorithm by Gagolewski, Bartoszuk, and Cena (2016). The Genie algorithm is based on the minimum spanning tree (MST) of the pairwise distance graph of a given point set. Just like the Single Linkage method, it consumes the edges of the MST in an increasing order of weights. However, it prevents the formation of clusters of highly imbalanced sizes; once the Gini index (see `gini_index()`) of the cluster size distribution raises above `gini_threshold`, merging a point group of the smallest size is enforced.

A clustering can also be computed with respect to the M-mutual reachability distance (based, e.g., on the Euclidean metric), which is used in the definition of the HDBSCAN* algorithm (see `mst()` for the definition). For the smoothing factor $M > 0$, outliers are pulled away from their neighbours. This way, the Genie algorithm gives an alternative to the HDBSCAN* algorithm (Campello et al., 2013) that is able to detect a predefined number of clusters and indicate outliers (via **deadwood**; see Gagolewski, 2026) without depending on DBSCAN*'s `eps` or HDBSCAN*'s `min_cluster_size` parameters. Also make sure to check out the Lumbermark algorithm (package **lumbermark**) that is also based on MSTs.

Usage

```
gclust(d, ...)

## Default S3 method:
gclust(
  d,
  gini_threshold = 0.3,
  M = 0L,
  distance = c("euclidean", "l2", "manhattan", "cityblock", "l1", "cosine"),
  verbose = FALSE,
  ...
)

## S3 method for class 'dist'
gclust(d, gini_threshold = 0.3, M = 0L, verbose = FALSE, ...)

## S3 method for class 'mst'
gclust(d, gini_threshold = 0.3, verbose = FALSE, ...)

genie(d, ...)

## Default S3 method:
genie(
  d,
  k,
  gini_threshold = 0.3,
  M = 0L,
  distance = c("euclidean", "l2", "manhattan", "cityblock", "l1", "cosine"),
  verbose = FALSE,
  ...
)

## S3 method for class 'dist'
genie(d, k, gini_threshold = 0.3, M = 0L, verbose = FALSE, ...)

## S3 method for class 'mst'
genie(d, k, gini_threshold = 0.3, verbose = FALSE, ...)
```

Arguments

d	a numeric matrix (or an object coercible to one, e.g., a data frame with numeric-like columns) or an object of class `dist` (see `dist`), or an object of class `mst` (see `mst`)
...	further arguments passed to `mst()`
gini_threshold	threshold for the Genie correction, i.e., the Gini index of the cluster size distribution; threshold of 1.0 leads to the single linkage algorithm; low thresholds highly penalise the formation of small clusters

M	smoothing factor; $M \leq 1$ gives the selected distance; otherwise, the mutual reachability distance is used
distance	metric used to compute the linkage, one of: "euclidean" (synonym: "l2"), "manhattan" (a.k.a. "l1" and "cityblock"), "cosine"
verbose	logical; whether to print diagnostic messages and progress information
k	the desired number of clusters to detect

Details

As with all distance-based methods (this includes k-means and DBSCAN as well), applying data preprocessing and feature engineering techniques (e.g., feature scaling, feature selection, dimensionality reduction) might lead to more meaningful results.

If d is a numeric matrix or an object of class dist, mst() will be called to compute an MST, which generally takes at most $O(n^2)$ time. However, by default, a faster algorithm based on K-d trees is selected automatically for low-dimensional Euclidean spaces; see mst_euclid from the **quitefastmst** package.

Once a minimum spanning tree is determined, the Genie algorithm runs in $O(n\sqrt{n})$ time. If you want to test different gini_thresholds or ks, it is best to compute the MST explicitly beforehand.

Due to Genie's original definition, the resulting partition tree (dendrogram) might violate the ultrametricity property (merges might occur at levels that are not increasing w.r.t. a between-cluster distance). gclust() automatically corrects departures from ultrametricity by applying height = rev(cummin(rev(height))).

Value

gclust() computes the entire clustering hierarchy; it returns a list of class hclust; see hclust. Use cutree to obtain an arbitrary k-partition.

genie() returns an object of class mstclust, which defines a k-partition, i.e., a vector whose i-th element denotes the i-th input point's cluster label between 1 and k.

In both cases, the mst attribute gives the computed minimum spanning tree which can be reused in further calls to the functions from **genieclust**, **lumbermark**, and **deadwood**. For genie(), the cut_edges attribute gives the $k-1$ indexes of the MST edges whose omission leads to the requested k-partition (connected components of the resulting spanning forest). In gclust(), these are exactly the last $k-1$ indexes in the links attribute (but sorted).

Author(s)

Marek Gagolewski and other contributors

References

M. Gagolewski, M. Bartoszuk, A. Cena, Genie: A new, fast, and outlier-resistant hierarchical clustering algorithm, *Information Sciences* 363, 2016, 8-23, doi:10.1016/j.ins.2016.05.003

R.J.G.B. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates, *Lecture Notes in Computer Science* 7819, 2013, 160-172, doi:10.1007/978364237456-2_14

M. Gagolewski, A. Cena, M. Bartoszuk, Ł. Brzozowski, Clustering with minimum spanning trees: How good can it be?, *Journal of Classification* 42, 2025, 90-112, doi:10.1007/s00357024094831

M. Gagolewski, genieclust: Fast and robust hierarchical clustering, *SoftwareX* 15, 2021, 100722, doi:10.1016/j.softx.2021.100722

M. Gagolewski, deadwood, in preparation, 2026

M. Gagolewski, quitefastmst, in preparation, 2026

See Also

The official online manual of **genieclust** at https://genieclust.gagolewski.com/

Gagolewski, M., **genieclust**: Fast and robust hierarchical clustering, *SoftwareX* 15:100722, 2021, doi:10.1016/j.softx.2021.100722

`mst()` for the minimum spanning tree routines

`normalized_clustering_accuracy()` (amongst others) for external cluster validity measures

Examples

```
library("datasets")
data("iris")
X <- jitter(as.matrix(iris[3:4]))
h <- gclust(X)
y_pred <- cutree(h, 3)
y_test <- as.integer(iris[,5])
plot(X, col=y_pred, pch=y_test, asp=1, las=1)
adjusted_rand_score(y_test, y_pred)
normalized_clustering_accuracy(y_test, y_pred)

# detect 3 clusters and find outliers with Deadwood
library("deadwood")
y_pred2 <- genie(X, k=3, M=5)  # the 5-mutual reachability distance
plot(X, col=y_pred2, asp=1, las=1)
is_outlier <- deadwood(y_pred2)
points(X[!is_outlier, ], col=y_pred2[!is_outlier], pch=16)
```

`inequality` *Inequality Measures*

Description

`gini_index()` gives the normalised Gini index, `bonferroni_index()` implements the Bonferroni index, and `devergottini_index()` implements the De Vergottini index.

Usage

```
gini_index(x)

bonferroni_index(x)

devergottini_index(x)
```

Arguments

x	numeric vector of non-negative values

Details

These indices can be used to quantify the "inequality" of a sample. They can be conceived as normalised measures of data dispersion. For constant vectors (perfect equity), the indices yield values of 0. Vectors with all elements but one equal to 0 (perfect inequality), are assigned scores of 1. They follow the Pigou-Dalton principle (are Schur-convex): setting $x_i = x_i - h$ and $x_j = x_j + h$ with $h > 0$ and $x_i - h \geq x_j + h$ (taking from the "rich" and giving to the "poor") decreases the inequality.

These indices have applications in economics, amongst others. The Genie clustering algorithm uses the Gini index as a measure of the inequality of cluster sizes.

The normalised Gini index is given by:

$$G(x_1, \ldots, x_n) = \frac{\sum_{i=1}^{n}(n - 2i + 1)x_{\sigma(n-i+1)}}{(n-1)\sum_{i=1}^{n} x_i}.$$

The normalised Bonferroni index is given by:

$$B(x_1, \ldots, x_n) = \frac{\sum_{i=1}^{n}(n - \sum_{j=1}^{i} \frac{n}{n-j+1})x_{\sigma(n-i+1)}}{(n-1)\sum_{i=1}^{n} x_i}.$$

The normalised De Vergottini index is given by:

$$V(x_1, \ldots, x_n) = \frac{1}{\sum_{i=2}^{n} \frac{1}{i}} \left(\frac{\sum_{i=1}^{n} \left(\sum_{j=i}^{n} \frac{1}{j} \right) x_{\sigma(n-i+1)}}{\sum_{i=1}^{n} x_i} - 1 \right).$$

Here, σ is an ordering permutation of $(x_1, \ldots, x_n)$.

Value

The value of the inequality index, a number in $[0, 1]$.

Author(s)

Marek Gagolewski and other contributors

References

Bonferroni, C., *Elementi di Statistica Generale*, Libreria Seber, Firenze, 1930.

Gini, C., *Variabilita e Mutabilita*, Tipografia di Paolo Cuppini, Bologna, 1912.

See Also

The official online manual of **genieclust** at https://genieclust.gagolewski.com/

Gagolewski, M., **genieclust**: Fast and robust hierarchical clustering, *SoftwareX* 15:100722, 2021, doi:10.1016/j.softx.2021.100722

Examples

```
gini_index(c(2, 2, 2, 2, 2))   # no inequality
gini_index(c(0, 0, 10, 0, 0))  # one has it all
gini_index(c(7, 0, 3, 0, 0))   # give to the poor, take away from the rich
gini_index(c(6, 0, 3, 1, 0))   # (a.k.a. the Pigou-Dalton principle)
bonferroni_index(c(2, 2, 2, 2, 2))
bonferroni_index(c(0, 0, 10, 0, 0))
bonferroni_index(c(7, 0, 3, 0, 0))
bonferroni_index(c(6, 0, 3, 1, 0))
devergottini_index(c(2, 2, 2, 2, 2))
devergottini_index(c(0, 0, 10, 0, 0))
devergottini_index(c(7, 0, 3, 0, 0))
devergottini_index(c(6, 0, 3, 1, 0))
```

Index