# Package 'genieclust'

July 24, 2025

**Type** Package

**Title** Fast and Robust Hierarchical Clustering with Noise Point Detection

**Version** 1.2.0

**Date** 2025-07-24

**Description** The Genie algorithm (Gagolewski, 2021 <DOI:10.1016/j.softx.2021.100722>) is a robust and outlier-resistant hierarchical clustering method (Gagolewski, Bartoszuk, Cena, 2016 <DOI:10.1016/j.ins.2016.05.003>). This package features its faster and more powerful version. It allows clustering with respect to mutual reachability distances, enabling it to act as a noise point detector or a version of 'HDBSCAN*' that can identify a predefined number of clusters. The package also features an implementation of 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, quitefastmst

**Suggests** datasets,

**LinkingTo** Rcpp

**Encoding** UTF-8

**SystemRequirements** OpenMP, C++17

**RoxygenNote** 7.3.2

**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** 2025-07-24 16:00:02 UTC

# Contents

cluster_validity . . . . . 2
compare_partitions . . . . . 4
gclust . . . . . 7
inequality . . . . . 11
mst . . . . . 13

**Index** **16**

---

`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 $L$ columns, $\sigma$ is a permutation of the set $\{1, \ldots, \max(K, L)\}$,

and $c_{i,\cdot} = c_{i,1} + ... + c_{i,L}$ is the i-th row sum, under the assumption that $c_{i,j} = 0$ for $i > K$ or $j > L$ and $0/0 = 0$.

`normalized_pivoted_accuracy()` is defined as $(\max_\sigma \sum_{j=1}^{\max(K,L)} \frac{c_{\sigma(j),j}/n - 1/\max(K,L)}{1 - 1/\max(K,L)}$, where $\sigma$ is a permutation of the set $\{1, \ldots, \max(K, L)\}$, and $n$ is the sum of all elements in $C$. For non-square matrices, missing rows/columns are assumed to be filled with 0s.

`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, 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`, the merging of a point group of the smallest size is enforced.

The clustering can also be computed with respect to the mutual reachability distances (based, e.g., on the Euclidean metric), which is used in the definition of the HDBSCAN* algorithm (see Campello et al., 2013). If $M > 1$, then the mutual reachability distance $m(i, j)$ with a smoothing factor $M$ is used instead of the chosen "raw" distance $d(i, j)$. It holds $m(i, j) = \max(d(i, j), c(i), c(j))$, where the core distance $c(i)$ is the distance to the $i$-th point's $(M - 1)$-th nearest neighbour. This makes "noise" and "boundary" points being more "pulled away" from each other.

The Genie correction together with the smoothing factor $M > 1$ (note that $M = 2$ corresponds to the original distance) gives a version of the HDBSCAN* algorithm that is able to detect a predefined number of clusters. Hence it does not dependent on the DBSCAN's `eps` parameter or the HDBSCAN's `min_cluster_size` one.

## Usage

```
gclust(d, ...)

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

## S3 method for class 'dist'
gclust(d, gini_threshold = 0.3, 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,
  distance = c("euclidean", "l2", "manhattan", "cityblock", "l1", "cosine"),
  M = 1L,
  postprocess = c("boundary", "none", "all"),
  detect_noise = M > 1L,
  verbose = FALSE,
  ...
)

## S3 method for class 'dist'
genie(
  d,
  k,
  gini_threshold = 0.3,
  M = 1L,
  postprocess = c("boundary", "none", "all"),
  detect_noise = M > 1L,
  verbose = FALSE,
  ...
```

```
)

## S3 method for class 'mst'
genie(
  d,
  k,
  gini_threshold = 0.3,
  postprocess = c("boundary", "none", "all"),
  detect_noise = FALSE,
  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 (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 |
| 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, $k = 1$ with $M > 1$ acts as a noise point detector |
| M | smoothing factor; $M \leq 2$ gives the selected distance; otherwise, the mutual reachability distance is used |
| postprocess | one of "boundary" (default), "none" or "all"; in effect only if $M > 1$. By default, only "boundary" points are merged with their nearest "core" points (A point is a boundary point if it is a noise point and it is amongst its adjacent vertex's $(M-1)$-th nearest neighbours). To force a classical k-partition of a data set (with no notion of noise), choose "all" |
| detect_noise | whether the minimum spanning tree's leaves should be marked as noise points, defaults to TRUE if $M > 1$ for compatibility with HDBSCAN* |

## Details

As in the case of all the distance-based methods, the standardisation of the input features is definitely worth giving a try. Oftentimes, more sophisticated feature engineering (e.g., dimensionality reduction) will 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 first explicitly.

According to the algorithm'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 a `k`-partition - a vector whose i-th element denotes the i-th input point's cluster label between 1 and `k` If `detect_noise` is `TRUE`, missing values (`NA`) denote noise points.

## Author(s)

Marek Gagolewski and other contributors

## References

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

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

Gagolewski, M., Cena, A., Bartoszuk, M., Brzozowski, L., Clustering with minimum spanning trees: How good can it be?, *Journal of Classification* 42, 2025, 90-112, doi:10.1007/s00357024-094831

## 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[2:3]))
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)
```

```
y_pred2 <- genie(X, 3, M=5)  # clustering wrt 5-mutual reachability distance
plot(X[,1], X[,2], col=y_pred2, pch=y_test, asp=1, las=1)
noise <- is.na(y_pred2)  # noise/boundary points
points(X[noise, ], col="gray", pch=10)
normalized_clustering_accuracy(y_test[!noise], y_pred2[!noise])
```

---

`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, $\sigma$ 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))
```

`mst` *Minimum Spanning Tree of the Pairwise Distance Graph*

## Description

Determine a(*) minimum spanning tree (MST) of the complete undirected graph representing a set of $n$ points whose weights correspond to the pairwise distances between the points.

## Usage

```
mst(d, ...)

## Default S3 method:
mst(
  d,
  distance = c("euclidean", "l2", "manhattan", "cityblock", "l1", "cosine"),
  M = 1L,
  verbose = FALSE,
  ...
)

## S3 method for class 'dist'
mst(d, M = 1L, verbose = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `d` | either 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` |
| `...` | further arguments passed to or from other methods, in particular, to `mst_euclid` from the **quitefastmst** package |
| `distance` | metric used in the case where `d` is a matrix; one of: `"euclidean"` (synonym: `"l2"`), `"manhattan"` (a.k.a. `"l1"` and `"cityblock"`), `"cosine"` |
| `M` | smoothing factor; `M` = 1 selects the requested `distance`; otherwise, the corresponding degree-`M` mutual reachability distance is used; `M` should be rather small, say, $\leq 20$ |
| `verbose` | logical; whether to print diagnostic messages and progress information |

## Details

(*) Note that if the distances are non unique, there might be multiple minimum trees spanning a given graph.

If `d` is a matrix and the use of Euclidean distance is requested (the default), then `mst_euclid` is called to determine the MST. It is quite fast in spaces of low intrinsic dimensionality, even for 10M points.

Otherwise, a much slower implementation of the Jarník (Prim/Dijkstra)-like method, which requires $O(n^2)$ time, is used. The algorithm is parallelised; the number of threads is determined by the

OMP_NUM_THREADS environment variable. As a rule of thumb, datasets up to 100k points should be processed relatively quickly.

If $M > 1$, then the mutual reachability distance $m(i, j)$ with the smoothing factor $M$ (see Campello et al. 2013) is used instead of the chosen "raw" distance $d(i, j)$. It holds $m(i, j) = \max\{d(i, j), c(i), c(j)\}$, where $c(i)$ is the core distance, i.e., the distance between the $i$-th point and its (M-1)-th nearest neighbour. This makes "noise" and "boundary" points being "pulled away" from each other. The Genie clustering algorithm (see gclust) with respect to the mutual reachability distance can mark some observations as noise points.

## Value

Returns a numeric matrix of class mst with $n-1$ rows and three columns: from, to, and dist sorted nondecreasingly. Its i-th row specifies the i-th edge of the MST which is incident to the vertices from[i] and to[i] with from[i] < to[i] (in 1,...,n) and dist[i] gives the corresponding weight, i.e., the distance between the point pair.

The Size attribute specifies the number of points, $n$. The Labels attribute gives the labels of the input points, if available. The method attribute provides the name of the distance function used.

If $M > 1$, the nn.index attribute gives the indices of the M-1 nearest neighbours of each point and nn.dist provides the corresponding distances, both in the form of an $n$ by $M - 1$ matrix.

## Author(s)

Marek Gagolewski and other contributors

## References

V. Jarník, O jistem problemu minimalnim, *Prace Moravske Prirodovedecke Spolecnosti* 6, 1930, 57-63.

C.F. Olson, Parallel algorithms for hierarchical clustering, *Parallel Computing* 21, 1995, 1313-1325.

R. Prim, Shortest connection networks and some generalisations, *The Bell System Technical Journal* 36(6), 1957, 1389-1401.

O. Borůvka, O jistém problému minimálním, *Práce Moravské Přírodovědecké Společnosti* 3, 1926, 37–58.

J.L. Bentley, Multidimensional binary search trees used for associative searching, *Communications of the ACM* 18(9), 509–517, 1975, doi:10.1145/361002.361007. W.B. March, R. Parikshit, A.G. Gray, Fast Euclidean minimum spanning tree: Algorithm, analysis, and applications, *Proc. 16th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining (KDD '10)*, 2010, 603–612.

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:10.1007/9783642374562_14.

## 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_euclid

## Examples

```
library("datasets")
data("iris")
X <- jitter(as.matrix(iris[1:2]))  # some data
T <- mst(X)
plot(X, asp=1, las=1)
segments(X[T[, 1], 1], X[T[, 1], 2],
         X[T[, 2], 1], X[T[, 2], 2])
```

# Index


adjusted_fm_score *(*compare_partitions*)*, 4
adjusted_mi_score *(*compare_partitions*)*, 4
adjusted_rand_score *(*compare_partitions*)*, 4

bonferroni_index *(*inequality*)*, 11

calinski_harabasz_index *(*cluster_validity*)*, 2
cluster_validity, 2
compare_partitions, 4
cutree, *10*

devergottini_index *(*inequality*)*, 11
dist, *9*, *13*
dunnowa_index *(*cluster_validity*)*, 2

fm_score *(*compare_partitions*)*, 4

gclust, 7, *14*
generalised_dunn_index *(*cluster_validity*)*, 2
genie *(*gclust*)*, 7
gini_index, *7*
gini_index *(*inequality*)*, 11

hclust, *10*

inequality, 11

mi_score *(*compare_partitions*)*, 4
mst, *9*, *10*, 13
mst_euclid, *9*, *13*, *15*

negated_ball_hall_index *(*cluster_validity*)*, 2
negated_davies_bouldin_index *(*cluster_validity*)*, 2
negated_wcss_index *(*cluster_validity*)*, 2
normalized_clustering_accuracy, *10*
normalized_clustering_accuracy *(*compare_partitions*)*, 4
normalized_confusion_matrix, *5*
normalized_confusion_matrix *(*compare_partitions*)*, 4
normalized_mi_score *(*compare_partitions*)*, 4
normalized_pivoted_accuracy *(*compare_partitions*)*, 4
normalizing_permutation *(*compare_partitions*)*, 4

pair_sets_index *(*compare_partitions*)*, 4

rand_score *(*compare_partitions*)*, 4

silhouette_index *(*cluster_validity*)*, 2
silhouette_w_index *(*cluster_validity*)*, 2

table, *5*, *6*

wcnn_index *(*cluster_validity*)*, 2