Package: deadwood (via r-universe)

June 3, 2026

Type Package

Title Outlier Detection via Pruning Mutual Reachability Minimum Spanning Trees

Version 0.9.0.9002

Date 2026-06-03

Description Implements an anomaly detection algorithm based on a dataset's mutual reachability minimum spanning tree: 'deadwood' chops protruding tree segments and marks small debris as outliers; see Gagolewski (2026) <https://deadwood.gagolewski.com/>. More precisely, the use of a mutual reachability distance pulls peripheral points farther away from each other. Tree edges with weights beyond the detected elbow point are removed. All the resulting connected components whose sizes are smaller than a given threshold are deemed anomalous. The 'Python' version of 'deadwood' is available via 'PyPI'.

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

URL https://deadwood.gagolewski.com/, https://github.com/gagolews/deadwood

License AGPL-3

Imports Rcpp, quitefastmst

Suggests datasets,

LinkingTo Rcpp

Encoding UTF-8

SystemRequirements OpenMP

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-06-03 13:27:07 UTC

RemoteUrl https://github.com/gagolews/deadwood

RemoteRef HEAD

RemoteSha 2b322dc9ff09223f32fd4b622b159d62503a0ff3

Contents

---

deadwood *Deadwood: Outlier Detection via Pruning Mutual Reachability Minimum Spanning Trees*

Description

Deadwood is an anomaly detection algorithm based on a dataset's mutual reachability minimum spanning tree. It chops protruding tree segments and marks small debris as outliers.

More precisely, the use of a mutual reachability distance pulls peripheral points farther away from each other. Tree edges with weights beyond the detected elbow point are removed. All the resulting connected components whose sizes are smaller than a given threshold are deemed anomalous.

Usage

```
deadwood(d, ...)

## Default S3 method:
deadwood(
  d,
  M = 5L,
  contamination = NA_real_,
  max_debris_size = NA_real_,
  max_contamination = 0.5,
  ema_dt = 0.01,
  connected = FALSE,
  distance = c("euclidean", "l2", "manhattan", "cityblock", "l1", "cosine"),
  verbose = FALSE,
  ...
)

## S3 method for class 'dist'
deadwood(
  d,
  M = 5L,
  contamination = NA_real_,
  max_debris_size = NA_real_,
  max_contamination = 0.5,
  ema_dt = 0.01,
  connected = FALSE,
  verbose = FALSE,
```

```
  ...
)

## S3 method for class 'mstclust'
deadwood(
  d,
  contamination = NA_real_,
  max_debris_size = NA_real_,
  max_contamination = 0.5,
  ema_dt = 0.01,
  verbose = FALSE,
  ...
)

## S3 method for class 'mst'
deadwood(
  d,
  contamination = NA_real_,
  max_debris_size = NA_real_,
  max_contamination = 0.5,
  ema_dt = 0.01,
  connected = FALSE,
  cut_edges = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`d`	a numeric matrix with n rows and p columns (or an object coercible to one, e.g., a data frame with numeric-like columns), an object of class `dist` (see `dist`), an object of class `mstclust` (see **genieclust** and **lumbermark**), or an object of class `mst` (see `mst`)
`...`	further arguments passed to `mst`
`M`	smoothing factor; $M \leq 1$ gives the selected `distance`; otherwise, the mutual reachability distance based on the M-th nearest neighbours is used
`contamination`	single numeric value or `NA`; the estimated (approximate) proportion of outliers in the dataset; if `NA`, the contamination amount will be determined by identifying the most significant elbow point of the curve comprised of increasingly ordered tree edge weights smoothened with an exponential moving average
`max_debris_size`	single integer value or `NA`; the maximal size of the leftover connected components that will be considered outliers; if `NA`, $\sqrt{n}$ is assumed
`max_contamination`	single numeric value; maximal contamination level assumed when `contamination` is `NA`

`ema_dt`	single numeric value; controls the smoothing parameter $\alpha = 1 - \exp(-dt)$ of the exponential moving average (in edge length elbow point detection), $y_i = \alpha w_i + (1-\alpha) y_{i-1}$, $y_1 = d_1$
`connected`	should the output tree be connected? $k = 1$ only; prunes branches instead of chopping the tree into pieces
`distance`	metric used in the case where `d` is a matrix; one of: `"euclidean"` (synonym: `"l2"`), `"manhattan"` (a.k.a. `"l1"` and `"cityblock"`), `"cosine"`
`verbose`	logical; whether to print diagnostic messages and progress information
`cut_edges`	numeric vector or `NULL`; $k-1$ indexes of the tree edges whose omission lead to k connected components (clusters), where the outliers are to be sought independently; most frequently this is generated via **genieclust** or **lumbermark**

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, for low-dimensional Euclidean spaces, a faster algorithm based on K-d trees is selected automatically; see `mst_euclid` from the **quitefastmst** package.

Once the spanning tree with increasingly sorted edge weights is determined ($\Omega(n \log n)$-$O(n^2)$), the Deadwood algorithm runs in $O(n)$ time. Memory use is $O(n)$.

Value

A logical vector `y` of length n, where `y[i] == TRUE` means that the `i`-th observation is deemed to be an outlier.

The `mst` attribute gives the computed minimum spanning tree which can be reused in further calls to the functions from **genieclust**, **lumbermark**, and **deadwood**. `cut_edges` gives the `cut_edges` passed as argument. `contamination` gives the detected contamination levels in each cluster (which can be different from the observed proportion of outliers detected).

Author(s)

Marek Gagolewski

References

M. Gagolewski, deadwood, in preparation, 2026, TODO

V. Satopää, J. Albrecht, D. Irwin, B. Raghavan, Finding a "Kneedle" in a haystack: Detecting knee points in system behavior, In: 31st Intl. Conf. Distributed Computing Systems Workshops, 2011, 166-171, doi:10.1109/ICDCSW.2011.20

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

See Also

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

Examples

```
library("datasets")
data("iris")
X <- jitter(as.matrix(iris[1:2]))  # some data
is_outlier <- deadwood(X, M=5)
plot(X, col=c("#ff000066", "#55555555")[is_outlier+1],
    pch=c(16, 1)[is_outlier+1], asp=1, las=1)
```

---

kneedle *Knee/Elbow Point Detection*

Description

Finds the most significant knee/elbow using the Kneedle algorithm with exponential smoothing.

Usage

```
kneedle_increasing(x, convex = TRUE, dt = 0.01)
```

Arguments

x	data vector (increasing)
convex	whether the data in x are convex-ish (elbow detection) or not (knee lookup)
dt	controls the smoothing parameter $\alpha = 1-\exp(-dt)$ of the exponential moving average, $y_i = \alpha x_i + (1-\alpha)y_{i-1}$, $y_1 = x_1$

Value

Returns the index of the knee/elbow point; 1 if not found.

Author(s)

Marek Gagolewski

References

V. Satopää, J. Albrecht, D. Irwin, B. Raghavan, Finding a "Kneedle" in a haystack: Detecting knee points in system behavior, In: 31st Intl. Conf. Distributed Computing Systems Workshops, 2011, 166-171, doi:10.1109/ICDCSW.2011.20

See Also

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

`mst` *Euclidean and Mutual Reachability Minimum Spanning Trees*

Description

A Euclidean minimum spanning tree (MST) provides a computationally convenient representation of a dataset: the n points are connected via $n-1$ shortest segments. Provided that the dataset has been appropriately preprocessed so that the distances between the points are informative, an MST can be applied in outlier detection, clustering, density estimation, dimensionality reduction, and many other topological data analysis tasks.

Usage

```
mst(d, ...)

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

## S3 method for class 'dist'
mst(d, M = 0L, 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 `mst_euclid` from **quitefastmst**
`M`	smoothing factor; $M = 0$ selects the requested `distance`; otherwise, the corresponding degree-`M` mutual reachability distance is used
`distance`	metric used in the case where `d` is a matrix; one of: `"euclidean"` (synonym: `"l2"`), `"manhattan"` (a.k.a. `"l1"` and `"cityblock"`), `"cosine"`
`verbose`	logical; whether to print diagnostic messages and progress information

Details

If `d` is a matrix and the Euclidean distance is requested (the default), then the MST is computed via a call to `mst_euclid` from **quitefastmst**. It is efficient in low-dimensional spaces. Otherwise, a general-purpose implementation of the Jarník (Prim/Dijkstra)-like $O(n^2)$-time algorithm is called.

If $M > 0$, then the minimum spanning tree is computed with respect to a mutual reachability distance (Campello et al., 2013): $d_M(i,j) = \max(d(i,j), c_M(i), c_M(j))$, where $d(i,j)$ is an ordinary

distance and $c_M(i)$ is the core distance given by $d(i,k)$ with k being i's M-th nearest neighbour (not including self, unlike in Campello et al., 2013). This pulls outliers away from their neighbours.

If the distances are not unique, there might be multiple trees spanning a given graph that meet the minimality property.

Value

Returns a numeric matrix of class `mst` with $n-1$ rows and three columns: `from`, `to`, and `dist`. 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. Edges are ordered increasingly with respect to their weights.

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 $M > 0$, the `nn.index` attribute gives the indexes of the `M` nearest neighbours of each point and `nn.dist` provides the corresponding distances, both in the form of an n by M matrix.

Author(s)

Marek Gagolewski

References

V. Jarník, O jistem problemu minimalnim (z dopisu panu O. Borůvkovi), *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. 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:10.1007/978364237456-2_14

M. Gagolewski, quitefastmst, in preparation, 2026, TODO

See Also

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

`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