Package ‘DDoutlier’

May 30, 2018

Type Package

Title Distance & Density-Based Outlier Detection

Version 0.1.0

Author Jacob H. Madsen <jacob.madsen1@mail.com>

Maintainer Jacob H. Madsen <jacob.madsen1@mail.com>

Description Outlier detection in multidimensional domains. Implementation of notable distance and density-based outlier algorithms. Allows users to identify local outliers by comparing observations to their nearest neighbors, reverse nearest neighbors, shared neighbors or natural neighbors. For distance-based approaches, see Knorr, M., & Ng, R. T. (1997) <doi:10.1145/782010.782021>, Angiulli, F., & Pizzuti, C. (2002) <doi:10.1007/3-540-45681-3_2>, Hautamaki, V., & Ismo, K. (2004) <doi:10.1109/ICPR.2004.1334558> and Zhang, K., Hutter, M. & Jin, H. (2009) <doi:10.1007/978-3-642-01307-2_84>. For density-based approaches, see Tang, J., Chen, Z., Fu, A. W. C., & Cheung, D. W. (2002) <doi:10.1007/3-540-47887-6_53>, Jin, W., Tung, A. K. H., Han, J., & Wang, W. (2006) <doi:10.1007/11731139_68>, Schubert, E., Zimek, A. & Kriegel, H-P. (2014) <doi:10.1137/1.9781611973440.63>, Latecki, L., Lazarevic, A. & Prokrajac, D. (2007) <doi:10.1007/978-3-540-73499-4_6>, Papadimitriou, S., Gibbons, P. B., & Faloutsos, C. (2003) <doi:10.1109/ICDE.2003.1260802>, Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000) <doi:10.1145/342009.335388>, Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. (2009) <doi:10.1145/1645953.1646195>, Zhu, Q., Feng, Ji. & Huang, J. (2016) <doi:10.1016/j.patrec.
3-642-20847-8_23>.

URL https://github.com/jhmadsen/DDoutlier

Encoding UTF-8

LazyData true

Imports dbscan, proxy, pracma

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-30 13:24:41 UTC

R topics documented:

`COF` *Connectivity-based Outlier Factor (COF) algorithm*

Description

Function to calculate the connectivity-based outlier factor as an outlier score for observations. Suggested by Tang, J., Chen, Z., Fu, A. W. C., & Cheung, D. W. (2002)

Usage

```
COF(dataset, k = 5)
```

Arguments

`dataset`	The dataset for which observations have a COF score returned
`k`	The number of k-nearest neighbors to construct a SBN-path with, being the number of neighbors for each observation to compare chaining-distance with. k has to be smaller than the number of observations in dataset

Details

COF computes the connectivity-based outlier factor for observations, being the comparison of chaining-distances between observation subject to outlier scoring and neighboring observations. The COF function is useful for outlier detection in clustering and other multidimensional domains.

Value

A vector of COF scores for observations. The greater the COF, the greater outlierness

Author(s)

Jacob H. Madsen

References

Tang, J., Chen, Z., Fu, A. W. C., & Cheung, D. W. (2002). Enhancing Effectiveness of Outlier Detections for Low Density Patterns. In Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD). Taipei. pp. 535-548. DOI: 10.1007/3-540-47887-6_53

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional k
outlier_score <- COF(dataset=X, k=10)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

`DB` *Distance-based outlier detection based on user-given neighborhood size*

Description

Function to calculate how many observations are within a certain sized neighborhood as an outlier score. Outliers are classified according to a user-given threshold of observations to be within the neighborhood. Suggested by Knorr, M., & Ng, R. T. (1997)

Usage

```
DB(dataset, d = 1, fraction = 0.05)
```

Arguments

`dataset`	The dataset for which observations are classified as outliers/inliers
`d`	The radius of the neighborhood
`fraction`	The proportion of the number of observations to be within the neighborhood for observations to be classified as inliers. If the proportion of observations within the neighborhood is less than the given fraction, observations are classified as outliers

Details

DB computes a neighborhood for each observation given a radius (argument 'd') and returns the number of neighbors within the neighborhood. Observations are classified as inliers or outliers, based on a proportion (argument 'fraction') of observations to be within the neighborhood

Value

`neighbors`	The number of neighbors within the neighborhood
`classification`	Binary classification of observations as inlier or outlier

Author(s)

Jacob H. Madsen

References

Knorr, M., & Ng, R. T. (1997). A Unified Approach for Mining Outliers. In Conf. of the Centre for Advanced Studies on Collaborative Research (CASCON). Toronto, Canada. pp. 236-248. DOI: 10.1145/782010.782021

Examples

```
# Create dataset
X <- iris[,1:4]

# Classify observations
cls_observations <- DB(dataset=X, d=1, fraction=0.05)$classification

# Remove outliers from dataset
X <- X[cls_observations=='Inlier',]
```

`INFLO` *Influenced Outlierness (INFLO) algorithm*

Description

Function to calculate the influenced outlierness as an outlier score for observations. Suggested by Jin, W., Tung, A. K. H., Han, J., & Wang, W. (2006)

Usage

```
INFLO(dataset, k = 5)
```

Arguments

`dataset`	The dataset for which observations have an INFLO score returned
`k`	The number of reverse k-nearest neighbors to compare density with. k has to be smaller than the number of observations in dataset

Details

INFLO computes the influenced outlierness score for observations, being the comparison of density in neighborhood of observation subject to outlier scoring and density in the reverse neighborhood. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The INFLO function is useful for outlier detection in clustering and other multidimensional domains

Value

A vector of INFLO scores for observations. The greater the INFLO, the greater outlierness

Author(s)

Jacob H. Madsen

References

Jin, W., Tung, A. K. H., Han, J., & Wang, W. (2006). Ranking Outliers Using Symmetric Neighborhood Relationship. In Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD). Singapore. pp 577-593. DOI: 10.1007/11731139_68

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional k
outlier_score <- INFLO(dataset=X, k=10)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

KDEOS — *Kernel Density Estimation Outlier Score (KDEOS) algorithm with gaussian kernel*

Description

Function to calculate a density estimation as an outlier score for observations, over a range of k-nearest neighbors. Suggested by Schubert, E., Zimek, A. & Kriegel, H-P. (2014)

Usage

```
KDEOS(dataset, k_min = 5, k_max = 10, eps = NULL)
```

Arguments

`dataset`	The dataset for which observations have an KDEOS score returned
`k_min`	The k parameter starting the k-range
`k_max`	The k parameter ending the k-range. Has to be smaller than the number of observations in dataset and greater than or equal to k_min
`eps`	An optional minimum bandwidth. If eps is smaller than the mean reachability distance for observations, eps is used. Otherwise mean reachability distance is used as bandwidth

Details

KDEOS computes a kernel density estimation over a user-given range of k-nearest neighbors. The score is normalized between 0 and 1, such that observation with 1 has the lowest density estimation and greatest outlierness. A gaussian kernel is used for estimation with a bandwidth being the reachability distance for neighboring observations. If a lower user-given bandwidth is desired, putting more weight on outlying observations, eps has to be lower than the mean reachability distance for observations. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The KDEOS function is useful for outlier detection in clustering and other multidimensional domains

Value

A vector of KDEOS scores normalized between 1 and 0, with 1 being the greatest outlierness

Author(s)

Jacob H. Madsen

References

Schubert, E., Zimek, A. & Kriegel, H-P. (2014). Generalized Outlier Detection with Flexible Kernel Density Estimates. Proceedings of the 2014 SIAM International Conference on Data Mining. Philadelphia, USA. pp. 542-550. DOI: 10.1137/1.9781611973440.63

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional range of k's
outlier_score <- KDEOS(dataset=X, k_min=10, k_max=15)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

KNN_AGG *Aggregated k-nearest neighbors distance over different k's*

Description

Function to calculate aggregated distance to k-nearest neighbors over a range of k's, as an outlier score. Suggested by Angiulli, F., & Pizzuti, C. (2002)

Usage

```
KNN_AGG(dataset, k_min = 5, k_max = 10)
```

Arguments

`dataset`	The dataset for which observations have an aggregated k-nearest neighbors distance returned
`k_min`	The k parameter starting the k-range
`k_max`	The k parameter ending the k-range. Has to be smaller than the number of observations in dataset and greater than or equal to k_min

Details

KNN_AGG computes the aggregated distance to neighboring observations by aggregating the results from k_min-NN to k_max-NN, such that if k_min=1 and k_max=3, results from 1NN, 2NN and 3NN are aggregated. A kd-tree is used for kNN computation, using the kNN function() from the 'dbscan' package. The KNN_AGG function is useful for outlier detection in clustering and other multidimensional domains.

Value

A vector of aggregated distance for observations. The greater the distance, the greater outlierness

Author(s)

Jacob H. Madsen

References

Angiulli, F., & Pizzuti, C. (2002). Fast Outlier Detection in High Dimensional Spaces. In Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD). Helsinki, Finland. pp. 15-26. DOI: 10.1007/3-540-45681-3_2

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting a range of k's
outlier_score <- KNN_AGG(dataset=X, k_min=10, k_max=15)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

KNN_IN *In-degree for observations in a k-nearest neighbors graph*

Description

Function to calculate in-degree as an outlier score for observations, given a k-nearest neighbors graph. Suggested by Hautamaki, V., & Ismo, K. (2004)

Usage

```
KNN_IN(dataset, k = 5)
```

Arguments

dataset	The dataset for which observations have an in-degree returned
k	The number of k-nearest neighbors to construct a graph with. Has to be smaller than the number of observations in dataset

Details

KNN_IN computes the in-degree, being the number of reverse neighbors. For computing the in-degree, a k-nearest neighbors graph is computed. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The KNN_IN function is useful for outlier detection in clustering and other multidimensional domains.

Value

A vector of in-degree for observations. The smaller the in-degree, the greater outlierness

Author(s)

Jacob H. Madsen

References

Hautamaki, V., & Ismo, K. (2004). Outlier Detection Using k-Nearest Neighbour Graph. In International Conference on Pattern Recognition. Cambridge, UK. pp. 430-433. DOI: 10.1109/ICPR.2004.1334558

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional k
outlier_score <- KNN_IN(dataset=X, k=10)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = FALSE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

KNN_SUM *Sum of distance to k-nearest neighbors*

Description

Function to calculate sum of distance to k-nearest neighbors as an outlier score, based on a kd-tree

Usage

```
KNN_SUM(dataset, k=5)
```

Arguments

dataset	The dataset for which observations have a summed k-nearest neighbors distance returned
k	The number of k-nearest neighbors. k has to be smaller than the number of observations in dataset

Details

KNN_SUM computes the sum of distance to neighboring observations. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The KNN_SUM function is useful for outlier detection in clustering and other multidimensional domains.

Value

A vector of summed distance for observations. The greater distance, the greater outlierness

Author(s)

Jacob H. Madsen

Examples

```
# Create dataset and set an optional k
X <- iris[,1:4]
K <- 5

# Find outliers
outlier_score <- KNN_SUM(dataset=X, k=K)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

LDF *Local Density Factor (LDF) algorithm with gaussian kernel*

Description

Function to calculate a Local Density Estimate (LDE) and Local Density Factor (LDF), as an outlier score, with a gaussian kernel. Suggested by Latecki, L., Lazarevic, A. & Prokrajac, D. (2007)

Usage

```
LDF(dataset, k = 5, h = 1, c = 1)
```

Arguments

`dataset`	The dataset for which observations have an LDE and LDF score returned
`k`	The number of k-nearest neighbors to compare density estimation with. k has to be smaller than number of observations in dataset
`h`	User-given bandwidth for kernel functions. The greater the bandwidth, the smoother kernels and lesser weight are put on outliers. Default is 1
`c`	Scaling constant for comparison of LDE to neighboring observations. LDF is the comparison of average LDE for an observation and its neighboring observations. Thus, c=1 gives results in an LDF between 0 and 1, while c=0 can result in very large or infinite values of LDF. Default is 1

Details

LDF computes a kernel density estimation, called LDE, over a user-given number of k-nearest neighbors. The LDF score is the comparison of Local Density Estimate (LDE) for an observation to its neighboring observations. Naturally, if an observation has a greater LDE than its neighboring observations, it has no outlierness whereas an observation with smaller LDE than its neighboring observations has great outlierness. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The LDF function is useful for outlier detection in clustering and other multidimensional domains

Value

`LDE`	A vector of Local Density Estimate for observations. The greater the LDE, the greater centrality
`LDF`	A vector of Local Density Factor for observations. The greater the LDF, the greater the outlierness

Author(s)

Jacob H. Madsen

References

Latecki, L., Lazarevic, A. & Prokrajac, D. (2007). Outlier Detection with Kernel Density Functions. International Workshop on Machine Learning and Data Mining in Pattern Recognition: Machine Learning and Data Mining in Pattern Recognition. pp. 61-75. DOI: 10.1007/978-3-540-73499-4_6

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional range of k's
outlier_score <- LDF(dataset=X, k=10, h=2, c=1)$LDF

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

`LDOF` *Local Distance-based Outlier Factor (LDOF) algorithm*

Description

Function to calculate Local Distance-based Outlier Factor (LDOF) as an outlier score for observations. Suggested by Zhang, K., Hutter, M. & Jin, H. (2009)

Usage

```
LDOF(dataset, k = 5)
```

Arguments

`dataset`	The dataset for which observations have an LDOF score returned
`k`	The number of nearest neighbors to compare distances with

Details

LDOF computes distance for an observations to its to k-nearest neighbors and compare the distance with the average distances between the nearest neighbors. The LDOF function is useful for outlier detection in clustering and other multidimensional domains

Value

A vector of LDOF scores for observations. The greater the LDOF score, the greater outlierness

Author(s)

Jacob H. Madsen

References

Zhang, K., Hutter, M. & Jin, H. (2009). A New Local Distance-based Outlier Detection Approach for Scattered Real-World Data. Pacific-Asia Conference on Knowledge Discovery and Data Mining: Advances in Knowledge Discovery and Data Mining. pp. 813-822. DOI: 10.1007/978-3-642-01307-2_84

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional range of k's
outlier_score <- LDOF(dataset=X, k=10)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

`LOCI` *Local Correlation Integral (LOCI) algorithm with constant nearest neighbor parameter*

Description

Function to calculate Local Correlation Integral (LOCI) as an outlier score for observations. Suggested by Papadimitriou, S., Gibbons, P. B., & Faloutsos, C. (2003). Uses a k number of nearest neighbors instead of a constant radius

Usage

```
LOCI(dataset, alpha = 0.5, nn = 20, k = 3)
```

Arguments

`dataset`	The dataset for which observations have a LOCI returned
`alpha`	The parameter setting the size of the sampling neighborhood, as a proportion of the counting neighborhood, for observations to identify other observations in their respective neighborhood. An alpha of 1 equals a sampling neighborhood the size of the counting neighborhood (the size of distance to nn). An alpha of 0.5 equals a sampling neighborhood half the size of the counting neighborhood
`nn`	The number of nearest neighbors to compare sampling neighborhood with. Original paper suggest a constant user-given radius that includes at least 20 neighbors in order to introduce statistical errors in MDEF. Default is 20
`k`	The number of standard deviations the sampling neighborhood of an observation should differ from the sampling neighborhood of neighboring observations, to be an outlier. Default is set to 3 as used in original papers experiments

Details

LOCI computes a counting neighborhood to the nn nearest observations, where the radius is equal to the outermost observation. Within the counting neighborhood each observation has a sampling neighborhood of which the size is determined by the alpha input parameter. LOCI returns an outlier score based on the standard deviation of the sampling neighborhood, called the multi-granularity deviation factor (MDEF). The LOCI function is useful for outlier detection in clustering and other multidimensional domains

Value

`npar_pi`	A vector of the number of observations within the sample neighborhood for observations
`avg_npar`	A vector of average number of observations within the sample neighborhood for neighboring observations
`sd_npar`	A vector of standard deviations for observations sample neighborhood

MDEF	A vector of the multi-granularity deviation factor (MDEF) for observations. The greater the MDEF, the greater the outlierness
norm_MDEF	A vector of normalized MDEF-values, being sd_npar/avg_npar
class	Classification of observations as inliers/outliers following the rule of k

Author(s)

Jacob H. Madsen

References

Papadimitriou, S., Gibbons, P. B., & Faloutsos, C. (2003). LOCI: Fast Outlier Detection Using the Local Correlation Integral. In International Conference on Data Engineering. pp. 315-326. DOI: 10.1109/ICDE.2003.1260802

Examples

```
# Create dataset
X <- iris[,1:4]

# Classify observations
cls_observations <- LOCI(dataset=X, alpha=0.5, nn=20, k=1)$class

# Remove outliers from dataset
X <- X[cls_observations=='Inlier',]
```

LOF *Local Outlier Factor (LOF) algorithm*

Description

Function to calculate the Local Outlier Factor (LOF) as an outlier score for observations. Suggested by Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000)

Usage

```
LOF(dataset, k = 5)
```

Arguments

dataset	The dataset for which observations have an LOF score returned
k	The number of k-nearest neighbors to compare density with. k has to be smaller than number of observations in dataset

Details

LOF computes a local density for observations with a user-given k-nearest neighbors. The density is compared to the density of the respective nearest neighbors, resulting in the local outlier factor. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The LOF function is useful for outlier detection in clustering and other multidimensional domains

Value

A vector of LOF scores for observations. The greater the LOF, the greater outlierness

Author(s)

Jacob H. Madsen

References

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. In Int. Conf. On Management of Data. Dallas, TX. pp. 93-104. DOI: 10.1145/342009.335388

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional k
outlier_score <- LOF(dataset=X, k=10)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

LOOP *Local Outlier Probability (LOOP) algorithm*

Description

Function to calculate the Local Outlier Probability (LOOP) as an outlier score for observations. Suggested by Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. (2009)

Usage

```
LOOP(dataset, k = 5, lambda = 3)
```

Arguments

`dataset`	The dataset for which observations have a LOOP score returned
`k`	The number of k-nearest neighbors to compare density with
`lambda`	Multiplication factor for standard deviation. The greater lambda, the smoother results. Default is 3 as used in original papers experiments

Details

LOOP computes a local density based on probabilistic set distance for observations, with a user-given k-nearest neighbors. The density is compared to the density of the respective nearest neighbors, resulting in the local outlier probability. The values ranges from 0 to 1, with 1 being the greatest outlierness. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The LOOP function is useful for outlier detection in clustering and other multidimensional domains

Value

A vector of LOOP scores for observations. 1 indicates outlierness and 0 indicate inlierness

Author(s)

Jacob H. Madsen

References

Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. (2009). LoOP: Local Outlier Probabilities. In ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China. pp. 1649-1652. DOI: 10.1145/1645953.1646195

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional k
outlier_score <- LOOP(dataset=X, k=10, lambda=3)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

NAN *Natural Neighbor (NAN) algorithm to return the self-adaptive neighborhood*

Description

Function to identify natural neighbors and the right k-parameter for kNN graphs as suggested by Zhu, Q., Feng, Ji. & Huang, J. (2016)

Usage

```
NAN(dataset, NaN_Edges = FALSE)
```

Arguments

`dataset`	The dataset for which natural neighbors are identified along with a k-parameter
`NaN_Edges`	Choice for computing natural neighbors. Computational heavy to compute

Details

NAN computes the natural neighbor eigenvalue and identifies natural neighbors in a dataset. The natural neighbor eigenvalue is powerful as k-parameter for computing a k-nearest neighborhood, being suitable for outlier detection, clustering or predictive modelling. Natural neighbors are defined as two observations being mutual k-nearest neighbors. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package

Value

`NaN_Num`	The number of in-degrees for observations given r
`r`	Natural neighbor eigenvalue. Useful as k-parameter
`NaN_Edges`	Matrix of edges for natural neighbors
`n_NaN`	The number of natural neighbors

Author(s)

Jacob H. Madsen

References

Zhu, Q., Feng, Ji. & Huang, J. (2016). Natural neighbor: A self-adaptive neighborhood method without parameter K. Pattern Recognition Letters. pp. 30-36. DOI: 10.1016/j.patrec.2016.05.007

Examples

```
# Select dataset
X <- iris[,1:4]

# Identify the right k-parameter
K <- NAN(X, NaN_Edges=FALSE)$r

# Use the k-setting in an abitrary outlier detection algorithm
outlier_score <- LOF(dataset=X, k=K)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

`NOF` *Natural Outlier Factor (NOF) algorithm*

Description

Function to calculate the Natural Outlier Factor (NOF) as an outlier score for observations. Suggested by Huang, J., Zhu, Q., Yang, L. & Feng, J. (2015)

Usage

```
NOF(dataset)
```

Arguments

`dataset`	The dataset for which observations have a NOF score returned

Details

NOF computes the nearest and reverse nearest neighborhood for observations, based on the natural neighborhood algorithm. Density is compared between observations and their neighbors. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package

Value

`nb`	A vector of in-degrees for observations
`max_nb`	Maximum in-degree observations in nb vector. Used as k-parameter in outlier detection of NOF
`r`	The natural neighbor eigenvalue
`NOF`	A vector of Natural Outlier Factor scores. The greater the NOF, the greater the outlierness

Author(s)

Jacob H. Madsen

References

Huang, J., Zhu, Q., Yang, L. & Feng, J. (2015). A non-parameter outlier detection algorithm based on Natural Neighbor. Knowledge-Based Systems. pp. 71-77. DOI: 10.1016/j.knosys.2015.10.014

Examples

```
# Select dataset
X <- iris[,1:4]

# Run NOF algorithm
outlier_score <- NOF(dataset=X)$NOF

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

RDOS — *Relative Density-based Outlier Factor (RDOS) algorithm with gaussian kernel*

Description

Function to calculate the Relative Density-based Outlier Factor (RDOS) as an outlier score for observations. Suggested by Tang, B. & Haibo, He. (2017)

Usage

```
RDOS(dataset, k = 5, h = 1)
```

Arguments

`dataset`	The dataset for which observations have an RDOS score returned
`k`	The number of k-nearest neighbors used to identify reverse- and shared nearest neighbors
`h`	Bandwidth parameter for gaussian kernel. A small h put more weight on outlying observations

Details

RDOS computes a kernel density estimation by combining the nearest, reverse nearest and shared neighbors into one neighborhood. The density estimation is compared to the density estimation of the neighborhoods observations. A gaussian kernel is used for density estimation, given a bandwidth chosen by user. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package.

It is a computational heavy task to identify reverse and shared neighbors from the kd-tree. Thus, the RDOS has high complexity and is not recommended to apply to datasets with n>5000. The RDOS function is useful for outlier detection in clustering and other multidimensional domains

Value

A vector of RDOS scores for observations. The greater the RDOS score, the greater outlierness

Author(s)

Jacob H. Madsen

References

Tang, B. & Haibo, He. (2017). A local density-based approach for outlier detection. Neurocomputing. pp. 171-180. DOI: 10.1016/j.neucom.2017.02.039

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional k
outlier_score <- RDOS(dataset=X, k=10, h=2)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

RKOF *Robust Kernel-based Outlier Factor (RKOF) algorithm with gaussian kernel*

Description

Function to to calculate the RKOF score for observations as suggested by Gao, J., Hu, W., Zhang, X. & Wu, Ou. (2011)

Usage

```
RKOF(dataset, k = 5, C = 1, alpha = 1, sigma2 = 1)
```

Arguments

`dataset`	The dataset for which observations have an RKOF score returned
`k`	The number of nearest neighbors to compare density estimation with
`C`	Multiplication parameter for k-distance of neighboring observations. Act as bandwidth increaser. Default is 1 such that k-distance is used for the gaussian kernel
`alpha`	Sensivity parameter for k-distance/bandwidth. Small alpha creates small variance in RKOF and vice versa. Default is 1
`sigma2`	Variance parameter for weighting of neighboring observations

Details

RKOF computes a kernel density estimation by comparing density estimation to the density of neighboring observations. A gaussian kernel is used for density estimation, given a bandwidth with k-distance. K-distance can be influenced with the parameters C and alpha. A kd-tree is used for kNN computation, using the kNN() function from the 'dbscan' package. The RKOF function is useful for outlier detection in clustering and other multidimensional domains

Value

A vector of RKOF scores for observations. The greater the RKOF score, the greater outlierness

Author(s)

Jacob H. Madsen

References

Gao, J., Hu, W., Zhang, X. & Wu, Ou. (2011). RKOF: Robust Kernel-Based Local Outlier Detection. Pacific-Asia Conference on Knowledge Discovery and Data Mining: Advances in Knowledge Discovery and Data Mining. pp. 270-283. DOI: 10.1007/978-3-642-20847-8_23

Examples

```
# Create dataset
X <- iris[,1:4]

# Find outliers by setting an optional k
outlier_score <- RKOF(dataset=X, k = 10, C = 1, alpha = 1, sigma2 = 1)

# Sort and find index for most outlying observations
names(outlier_score) <- 1:nrow(X)
sort(outlier_score, decreasing = TRUE)

# Inspect the distribution of outlier scores
hist(outlier_score)
```

Index