Package ‘clusterHD’

May 5, 2022

Type Package
Title Tools for Clustering High-Dimensional Data
Version 1.0.1
Date 2022-05-05
Author Jakob Raymaekers [aut, cre],
     Ruben Zamar [aut]
Maintainer Jakob Raymaekers <j.raymaekers@maastrichtuniversity.nl>
Description Tools for clustering high-dimensional data.
     In particular, it contains the methods described in
     <doi:10.1093/bioinformatics/btaa243>,
License GPL (>= 2)
Encoding UTF-8
RoxygenNote 7.1.2
Imports Rcpp (>= 1.0.7), stats, mclust, Ckmeans.1d.dp, cluster
LinkingTo Rcpp, RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-05-05 15:20:05 UTC

R topics documented:

  diagPlot .................................................. 2
  getLambda ................................................. 3
  HTKmeans .................................................. 4
  PVS ......................................................... 5

Index 8
diagPlot

diagnostic plots for HTK-Means Clustering

Description
Make diagnostic plots for HTK-means clustering.

Usage
diagPlot(HTKmeans.out, type = 1)

Arguments
- HTKmeans.out: the output of a call to HTKmeans.
- type: if type = 1, plots the regularization path. If type = 2, plots the differences in WCSS and ARI against the number of active variables.

Details
This visualization plots the regularization path or the differences in WCSS and ARI against the number of active variables.

Value
No return value, makes the plot directly.

Author(s)
J. Raymaekers and R.H. Zamar

References

See Also
HTKmeans

Examples
X <- iris[, -5]
lambdas <- seq(0, 1, by = 0.01)
HTKmeans.out <- HTKmeans(X, 3, lambdas)

diagPlot(HTKmeans.out, 1)
diagPlot(HTKmeans.out, 2)
Description

Select the regularization parameter for HTK-means clustering based on information criteria.

Usage

getLambda(HTKmeans.out, type = "AIC")

Arguments

HTKmeans.out the output of a call to HTKmeans.

type either "AIC" (default) or "BIC".

Details

This function selects the best lambda (based on information criteria AIC or BIC) out of the HTKmeans.out$inputargs$lambdas sequence of values.

Value

The selected value for lambda

Author(s)

J. Raymaekers and R.H. Zamar

References


See Also

HTKmeans

Examples

X <- mclust::banknote
y <- as.numeric(as.factor(X[, 1]))
lambdas <- seq(0, 1, by = 0.01)
X <- X[, -1]
HTKmeans.out <- HTKmeans(X, 2, lambdas)

# Both AIC and BIC suggest a lambda of 0.02 here:
getLambda(HTKmeans.out, "AIC")
getLambda(HTKmeans.out, "BIC")

HTKmeans

HTK-Means Clustering

Description
Perform HTK-means clustering (Raymaekers and Zamar, 2022) on a data matrix.

Usage
HTKmeans(X, k, lambdas,
standardize = TRUE,
iter.max = 100, nstart = 100)

Arguments
X a matrix containing the data.
k the number of clusters.
lambdas a vector of values for the regularization parameter lambda.
standardize logical flag for standardization to mean 0 and variance 1 of the data in X. This is recommended, unless the variance of the variables is known to quantify relevant information.
iter.max the maximum number of iterations allowed.
nstart number of starts used when k-means is applied to generate the starting values for HTK-means. See below for more info.

Details
The algorithm starts by generating a number of sparse starting values. This is done using k-means on subsets of variables. See Raymaekers and Zamar (2022) for details.

Value
A list with components:

- HTKmeans.out
  A list with length equal to the number of lambda values supplied in lambdas. Each element of this list is in turn a list containing centers A matrix of cluster centres. cluster A vector of integers (from 1:k) indicating the cluster to which each point is allocated. itnb The number of iterations executed until convergence converged Whether the algorithm stopped by converging or through reaching the maximum number of itertions.
- inputargs
  the input arguments to the function.
PVS

Pooled variable scaling for cluster analysis

Description

The function computes a scale for each variable in the data. The result can then be used to standardize a dataset before applying a clustering algorithm (such as k-means). The scale estimation is based on pooled scale estimators, which result from clustering the individual variables in the data. The method is proposed in Raymaekers, and Zamar (2020) <doi:10.1093/bioinformatics/btaa243>.

Usage

PVS(X, kmax = 3, dist = "euclidean", method = "gap", B = 1000, gapMethod = "firstSEmax", minSize = 0.05, rDist = runif, SE.factor = 1, refDist = NULL)

Arguments

X an n by p data matrix.

kmax maximum number of clusters in one variable. Default is 3.

dist "euclidean" for pooled standard deviation and "manhattan" for pooled mean absolute deviation. Default is "euclidean".

method either "gap" or "jump" to determine the number of clusters. Default is "gap".

B number of bootstrap samples for the reference distribution of the gap statistic. Default is 1000.
gapMethod method to define number of clusters in the gap statistic. See `cluster::maxSE` for more info. Defaults to "firstSEmax".

minSize minimum cluster size as a percentage of the total number of observations. Defaults to 0.05.

rDist Optional. Reference distribution (as a function) for the gap statistic. Defaults to `runif`, the uniform distribution.

SE.factor factor for determining number of clusters when using the gap statistic. See `cluster::maxSE` for more details. Defaults to 1

refDist Optional. A k by 2 matrix with the mean and standard error of the reference distribution of the gap statistic in its columns. Can be used to avoid bootstrapping when repeatedly applying the function to same size data.

Value
A vector of length p containing the estimated scales for the variables.

Author(s)
Jakob Raymaekers

References

Examples

```r
X <- iris[, -5]
y <- unclass(iris[, 5])

# Compute scales using different scale estimators.
# the pooled standard deviation is considerably smaller for variable 3 and 4:
sds <- apply(X, 2, sd); round(sds, 2)
ranges <- apply(X, 2, function(y) diff(range(y))); round(ranges, 2)
psds <- PVS(X); round(psds, 2)

# Now cluster using k-means after scaling the data
nbclus <- 3
kmeans.std <- kmeans(X, nbclus, nstart = 100) # no scaling
kmeans.sd <- kmeans(scale(X), nbclus, nstart = 100)
kmeans.rg <- kmeans(scale(X, scale = ranges), nbclus, nstart = 100)
kmeans.psd <- kmeans(scale(X, scale = psds), nbclus, nstart = 100)

# Calculate the Adjusted Rand Index for each of the clustering outcomes
round(mclust::adjustedRandIndex(y, kmeans.std$cluster), 2)
round(mclust::adjustedRandIndex(y, kmeans.sd$cluster), 2)
round(mclust::adjustedRandIndex(y, kmeans.rg$cluster), 2)
round(mclust::adjustedRandIndex(y, kmeans.psd$cluster), 2)
```
\begin{verbatim}
round(mclust::adjustedRandIndex(y, kmeans.psd$cluster), 2)
\end{verbatim}
Index

cluster::maxSE, 6
diagPlot, 2
getLambda, 3
HTKmeans, 2, 3, 4
kmeans, 5
PVS, 5