Package 'ksharp'

January 26, 2020

Title Cluster Sharpening

Version 0.1.0.1

Author Tomasz Konopka [aut, cre]

Maintainer Tomasz Konopka <tokonopka@gmail.com>

Description Clustering typically assigns data points into discrete groups, but the clusters can sometimes be indistinct. Cluster sharpening adjusts an existing clustering to create contrast between groups. This package provides a general interface for cluster sharpening along with several implementations based on different excision criteria.

Depends R (>= 3.5.0)

Imports methods, stats

License MIT + file LICENSE

URL https://github.com/tkonopka/ksharp

BugReports https://github.com/tkonopka/ksharp/issues

LazyData true

Suggests cluster, dbscan, knitr, Rcssplot (>= 1.0.0), rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-01-26 10:10:02 UTC

R topics documented:

`kdata.1` *Toy dataset with two convex groups with partial overlap*

Description

Toy dataset with two convex groups with partial overlap

Usage

```
data(kdata.1)
```

Format

matrix with two columns: `D1`, `D2`

`kdata.2` *Toy dataset with two non-overalpping and non-spherical groups*

Description

Toy dataset with two non-overalpping and non-spherical groups

Usage

```
data(kdata.2)
```

Format

matrix with two columns: `D1`, `D2`

kdata.3 *Toy dataset with three groups*

Description

Toy dataset with three groups

Usage

```
data(kdata.3)
```

Format

matrix with two columns: `D1`, `D2`

kdata.4 *Toy dataset with four groups atop a wide area of noise points*

Description

Toy dataset with four groups atop a wide area of noise points

Usage

```
data(kdata.4)
```

Format

matrix with two columns: `D1`, `D2`

ksharp *sharpen a clustering*

Description

Each data point in a clustering is assigned to a cluster, but some data points may lie in ambiguous zones between two or more clusters, or far from other points. Cluster sharpening assigns these border points into a separate noise group, thereby creating more stark distinctions between groups.

Usage

```
ksharp(
  x,
  threshold = 0.1,
  data = NULL,
  method = c("silhouette", "neighbor", "medoid"),
  threshold.abs = NULL
)
```

Arguments

x	clustering object; several types of inputs are acceptable, including objects of class kmeans, pam, and self-made lists with a component "cluster".
threshold	numeric; the fraction of points to place in noise group
data	matrix, raw data corresponding to clustering x; must be present when sharpening for the first time or if data is not present within x.
method	character, determines method used for sharpening
threshold.abs	numeric; absolute-value of threshold for sharpening. When non-NULL, this value overrides value in argument 'threshold'

Details

Noise points are assigned to a group with cluster index 0. This is analogous behavior to output produced by dbscan.

Value

clustering object based on input x, with adjusted cluster assignments and additional list components with sharpness measures. Cluster assignments are placed in $cluster and excised data points are given a cluster index of 0. Original cluster assignments are saved in $cluster.original. Sharpness measures are stored in components $silinfo, $medinfo, and $neiinfo, although these details may change in future versions of the package.

Examples

```
# prepare iris dataset for analysis
iris.data = iris[, 1:4]
rownames(iris.data) = paste0("iris_", seq_len(nrow(iris.data)))

# cluster the dataset into three groups
iris.clustered = kmeans(iris.data, centers=3)
table(iris.clustered$cluster)

# sharpen the clustering by excluding 10% of the data points
iris.sharp = ksharp(iris.clustered, threshold=0.1, data=iris.data)
table(iris.sharp$cluster)

# visualize cluster assignments
```

```
iris.pca = prcomp(iris.data)$x[,1:2]
plot(iris.pca, col=iris$Species, pch=ifelse(iris.sharp$cluster==0, 1, 19))
```

`medinfo` *compute info on distances to medoids/centroids*

Description

Analogous in structure to silinfo and neiinfo, it computes a "widths" matrix assessing how well each data point belongs to its cluster. Here, this measure is the ratio of two distances: in the numerator, the distance from the point to the nearest cluster center, and in the denominator, from the point to its own cluster center.

Usage

```
medinfo(cluster, data, silwidths)
```

Arguments

`cluster`	named vector
`data`	matrix with raw data
`silwidths`	matrix with silhouette widths

Value

list with component widths. The widths object is a matrix with one row per data item, with column med_ratio holding the sharpness measure.

Examples

```
# construct a manual clustering of the iris dataset
iris.data = iris[, 1:4]
rownames(iris.data) = paste0("iris_", seq_len(nrow(iris.data)))
iris.dist = dist(iris.data)
iris.clusters = setNames(as.integer(iris$Species), rownames(iris.data))

# compute sharpnessvalues based on medoids
iris.silinfo = silinfo(iris.clusters, iris.dist)
medinfo(iris.clusters, iris.data, iris.silinfo$widths)
```

`neiinfo` *Compute info on 'neighbor widths'*

Description

This function provides information on how well each data point belongs to its cluster. For each query point, the function considers n of its nearest neighbors. The neighbor widths are defined as the fraction of those neighbors that belong to the same cluster as the query point. These values are termed 'widths' in analogy to silhouette widths, another measure of cluster membership.

Usage

```
neiinfo(cluster, dist)
```

Arguments

`cluster`	vector with assignments of data elements to clusters
`dist`	distance object or matrix

Details

The function follows a similar signature as silinfo from this package.

Value

list with component widths. The wdiths object is a matrix with one row per data item, wth column neighborhood holding the sharpness value.

Examples

```
# construct a manual clustering of the iris dataset
iris.data = iris[, 1:4]
rownames(iris.data) = paste0("iris_", seq_len(nrow(iris)))
iris.dist = dist(iris.data)
iris.clusters = setNames(as.integer(iris$Species), rownames(iris.data))

# compute neighbor-based sharpness widths
neiinfo(iris.clusters, iris.dist)
```

`silinfo` *Compute info on silhouette widths*

Description

This function provides information on how well each data point belongs to its cluster. For each query point, the function considers the average distance to other members of the same cluster and the average distance to members of another, nearest, cluster. The widths are defined as the

Usage

```
silinfo(cluster, dist)
```

Arguments

`cluster`	vector with assignments of data elements to clusters
`dist`	distance object or matrix

Details

The function signature is very similar to cluster::silhouette but the implementation has important differences. This implementation requires both the dist object and and cluster vector must have names. This prevents accidental assignment of silhouette widths to the wrong elements.

Value

list, analogous to object within output from cluster::pam. In particular, the list has a component widths. The widths object is matrix with one row per data item, with column sil_width holding the silhouette width.

Examples

```
# construct a manual clustering of the iris dataset
iris.data = iris[, 1:4]
rownames(iris.data) = paste0("iris_", seq_len(nrow(iris.data)))
iris.dist = dist(iris.data)
iris.clusters = setNames(as.integer(iris$Species), rownames(iris.data))

# compute sharpness values based on silhouette widths
silinfo(iris.clusters, iris.dist)
```

Index