Package 'fastcluster'

August 21, 2017

Encoding UTF-8

Type Package

Version 1.1.24

Date 2017-08-14

Title Fast Hierarchical Clustering Routines for R and Python

Enhances stats, flashClust

Depends R (>= 3.0.0)

Description This is a two-in-one package which provides interfaces to both R and Python. It implements fast hierarchical, agglomerative clustering routines. Part of the functionality is designed as drop-in replacement for existing routines: linkage() in the SciPy package 'scipy.cluster.hierarchy', hclust() in R's 'stats' package, and the 'flashClust' package. It provides the same functionality with the benefit of a much faster implementation. Moreover, there are memory-saving routines for clustering of vector data, which go beyond what the existing packages provide. For information on how to install the Python files, see the file INSTALL in the source distribution.

License FreeBSD | GPL-2 | file LICENSE

URL http://danifold.net/fastcluster.html

NeedsCompilation yes

Author Daniel Müllner [aut, cph, cre]

Maintainer Daniel Müllner <daniel@danifold.net>

Repository CRAN

Date/Publication 2017-08-21 10:36:48 UTC

R topics documented:

`fastcluster` *Fast hierarchical, agglomerative clustering routines for R and Python*

Description

The **fastcluster** package provides efficient algorithms for hierarchical, agglomerative clustering. In addition to the R interface, there is also a Python interface to the underlying C++ library, to be found in the source distribution.

Details

The function `hclust` provides clustering when the input is a dissimilarity matrix. A dissimilarity matrix can be computed from vector data by `dist`. The `hclust` function can be used as a drop-in replacement for existing routines: `stats::hclust` and `flashClust::hclust` alias `flashClust::flashClust`. Once the fastcluster library is loaded at the beginning of the code, every program that uses hierarchical clustering can benefit immediately and effortlessly from the performance gain

When the package is loaded, it overwrites the function `hclust` with the new code.

The function `hclust.vector` provides memory-saving routines when the input is vector data.

Further information:

- R documentation pages: `hclust`, `hclust.vector`
- A comprehensive User's manual: fastcluster.pdf. Get this from the R command line with `vignette('fastcluster')`.
- JSS paper: `https://www.jstatsoft.org/v53/i09/`.
- See the author's home page for a performance comparison: `http://danifold.net/fastcluster.html`.

Author(s)

Daniel Müllner

References

`http://danifold.net/fastcluster.html`

See Also

`hclust`, `hclust.vector`

Examples

```
# Taken and modified from stats::hclust
#
# hclust(...)        # new method
# hclust.vector(...) # new method
# stats::hclust(...) # old method

require(fastcluster)
require(graphics)

hc <- hclust(dist(USArrests), "ave")
plot(hc)
plot(hc, hang = -1)

## Do the same with centroid clustering and squared Euclidean distance,
## cut the tree into ten clusters and reconstruct the upper part of the
## tree from the cluster centers.
hc <- hclust.vector(USArrests, "cen")
# squared Euclidean distances
hc$height <- hc$height^2
memb <- cutree(hc, k = 10)
cent <- NULL
for(k in 1:10){
  cent <- rbind(cent, colMeans(USArrests[memb == k, , drop = FALSE]))
}
hc1 <- hclust.vector(cent, method = "cen", members = table(memb))
# squared Euclidean distances
hc1$height <- hc1$height^2
opar <- par(mfrow = c(1, 2))
plot(hc,  labels = FALSE, hang = -1, main = "Original Tree")
plot(hc1, labels = FALSE, hang = -1, main = "Re-start from 10 clusters")
par(opar)
```

---

hclust *Fast hierarchical, agglomerative clustering of dissimilarity data*

Description

This function implements hierarchical clustering with the same interface as `hclust` from the **stats** package but with much faster algorithms.

Usage

```
hclust(d, method="complete", members=NULL)
```

Arguments

`d`	a dissimilarity structure as produced by `dist`.

method	the agglomeration method to be used. This must be (an unambiguous abbreviation of) one of "single", "complete", "average", "mcquitty", "ward.D", "ward.D2", "centroid" or "median".
members	NULL or a vector with length the number of observations.

Details

See the documentation of the original function hclust in the **stats** package.

A comprehensive User's manual fastcluster.pdf is available as a vignette. Get this from the R command line with vignette('fastcluster').

Value

An object of class 'hclust'. It encodes a stepwise dendrogram.

Author(s)

Daniel Müllner

References

http://danifold.net/fastcluster.html

See Also

fastcluster, hclust.vector, stats::hclust

Examples

```
# Taken and modified from stats::hclust
#
# hclust(...)        # new method
# stats::hclust(...) # old method

require(fastcluster)
require(graphics)

hc <- hclust(dist(USArrests), "ave")
plot(hc)
plot(hc, hang = -1)

## Do the same with centroid clustering and squared Euclidean distance,
## cut the tree into ten clusters and reconstruct the upper part of the
## tree from the cluster centers.
hc <- hclust(dist(USArrests)^2, "cen")
memb <- cutree(hc, k = 10)
cent <- NULL
for(k in 1:10){
  cent <- rbind(cent, colMeans(USArrests[memb == k, , drop = FALSE]))
}
hc1 <- hclust(dist(cent)^2, method = "cen", members = table(memb))
```

```
opar <- par(mfrow = c(1, 2))
plot(hc,  labels = FALSE, hang = -1, main = "Original Tree")
plot(hc1, labels = FALSE, hang = -1, main = "Re-start from 10 clusters")
par(opar)
```

`hclust.vector` *Fast hierarchical, agglomerative clustering of vector data*

Description

This function implements hierarchical, agglomerative clustering with memory-saving algorithms.

Usage

```
hclust.vector(X, method="single", members=NULL, metric='euclidean', p=NULL)
```

Arguments

`X`	an $(N \times D)$ matrix of 'double' values: N observations in D variables.
`method`	the agglomeration method to be used. This must be (an unambiguous abbreviation of) one of `"single"`, `"ward"`, `"centroid"` or `"median"`.
`members`	`NULL` or a vector with length the number of observations.
`metric`	the distance measure to be used. This must be one of `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"` or `"minkowski"`. Any unambiguous substring can be given.
`p`	parameter for the Minkowski metric.

Details

The function `hclust.vector` provides clustering when the input is vector data. It uses memory-saving algorithms which allow processing of larger data sets than `hclust` does.

The `"ward"`, `"centroid"` and `"median"` methods require `metric="euclidean"` and cluster the data set with respect to Euclidean distances.

For `"single"` linkage clustering, any dissimilarity measure may be chosen. Currently, the same metrics are implemented as the `dist` function provides.

The call

```
hclust.vector(X, method='single', metric=[...])
```

gives the same result as

```
hclust(dist(X, metric=[...]), method='single')
```

but uses less memory and is equally fast.

For the Euclidean methods, care must be taken since `hclust` expects **squared** Euclidean distances. Hence, the call

```
hclust.vector(X, method='centroid')
```

is, aside from the lesser memory requirements, equivalent to

```
d = dist(X)
hc = hclust(d^2, method='centroid')
hc$height = sqrt(hc$height)
```

The same applies to the "median" method. The "ward" method in hclust.vector is equivalent to hclust with method "ward.D2", but to method "ward.D" only after squaring as above.

More details are in the User's manual fastcluster.pdf, which is available as a vignette. Get this from the R command line with vignette('fastcluster').

Author(s)

Daniel Müllner

References

http://danifold.net/fastcluster.html

See Also

fastcluster, hclust

Examples

```
# Taken and modified from stats::hclust
## Perform centroid clustering with squared Euclidean distances,
## cut the tree into ten clusters and reconstruct the upper part of the
## tree from the cluster centers.
hc <- hclust.vector(USArrests, "cen")
# squared Euclidean distances
hc$height <- hc$height^2
memb <- cutree(hc, k = 10)
cent <- NULL
for(k in 1:10){
  cent <- rbind(cent, colMeans(USArrests[memb == k, , drop = FALSE]))
}
hc1 <- hclust.vector(cent, method = "cen", members = table(memb))
# squared Euclidean distances
hc1$height <- hc1$height^2
opar <- par(mfrow = c(1, 2))
plot(hc,  labels = FALSE, hang = -1, main = "Original Tree")
plot(hc1, labels = FALSE, hang = -1, main = "Re-start from 10 clusters")
par(opar)
```

Index