Package ‘clusterGenomics’

February 19, 2015

Type Package

Title Identifying clusters in genomics data by recursive partitioning

Version 1.0

Date 2013-07-01

Author Gro Nilsen and Ole Christian Lingjaerde

Maintainer Gro Nilsen <gronilse@ifi.uio.no>

Description The Partitioning Algorithm based on Recursive Thresholding
(PART) is used to recursively uncover clusters and subclusters
in the data. Functionality is also available for visualization
of the clustering.

Depends

License Artistic-2.0

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-07-02 19:51:19

R topics documented:

`exData1` *A simulated data set with 5 true clusters*

Description

A two-dimensional simulated data set which has 5 true clusters (the second data set in Figure 1 in the referenced paper).

Usage

```
data(exData1)
```

Format

A list with components `X` and `groups` where the former is a 65 x 2 data matrix and the latter is a vector of length 65 giving the groups label for each case (row) in `X`.

Source

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

Examples

```
#Get data
data(exData1)
```

`exData2` *A simulated data set with 4 true clusters*

Description

A 100-dimensional simulated data set which has 4 true clusters (a data set generated from simulation scenario E in the referenced paper).

Usage

```
data(exData2)
```

Format

A list with components `Y` and `groups` where the former is a 100 x 100 data matrix and the latter is a vector of length 100 giving the groups label for each case (row) in `Y`.

Source

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

Examples

```
#Get data
data(exData2)
```

`gap` *The gap statistic*

Description

Use the Gap statistic to determine the best number of clusters in a data set

Usage

```
gap(X,Kmax=10,B=100,ref.gen="PC",cl.lab=NULL,...)
```

Arguments

`X`	a numeric data matrix whose rows are to be clustered using a specified clustering algorithm (default is hierarchical clustering with average linkage and Euclidean distance, see below for other options)
`Kmax`	the maximum number of clusters to be evaluated.
`B`	the number of reference data sets to be generated in the calculation of the gap statistic.
`ref.gen`	a text string specifying how the reference data should be generated. Options are "PC" (reference data are generated uniformly over a box aligned with the principal components of the data) and "range" (reference data are generated uniformly over the range of the data). See the referenced paper for more details.
`cl.lab`	optional list of length `Kmax` giving vectors of cluster labels for the rows in `X` when partitioned into `1,..,Kmax` clusters.
`...`	other optional parameters including:

`cl.method`**:** the desired clustering method. Options currently include "hclust" (default) and "kmeans".

`linkage`**:** the desired linkage to be applied if `cl.method="hclust"`. Default is "average", see the parameter `method` in `hclust` for other options.

`dist.method`**:** the desired distance measure to be applied if `cl.method="hclust"`. Default is "euclidean". Other options include those supported by `dist` (under `method`), "sq.euclidean" (squared Euclidean distance) and "cor" (1 minus correlation distance).

`cor.method`**:** the correlation measure to be used if `dist.method="cor"`. Default is "pearson", see the parameter `method` in `cor` for other options.

`nstart`**:** the number of initial center sets to be applied if `cl.method="kmeans"`. Default is 10. See `kmeans` for details on this.

Details

The rows in X are partitioned into k = 1,..,Kmax clusters, and the Gap statistic is calculated for each partition. The best partition, and hence the best number of clusters, is selected using the Gap criterion (see the reference below).

Value

hatK	the best number of clusters according to the Gap criterion.
lab.hatK	a vector of same length as the number of rows in X assigning a group label to each case (row) in X based on the best partition as evaluated by Gap.
gap	a vector of length Kmax giving the Gap statistic for each evaluated partition.
sk	a vector of length Kmax giving the standard errors of the Gap statistics.
W	a vector of length Kmax giving the total within-cluster dispersion for each evaluated partition.

Author(s)

Gro Nilsen

References

Tibshirani et al., "Estimating the number of clusters in a data set via the gap statistic", Journal of the Royal Statistical Society B, 63, 2001

Examples

```
#Load a simulated data set with 5 clusters
data(exData1)
X = exData1$X
groups = exData1$groups

#Run gap (limit the number of reference data sets to decrease computing time):
res <- gap(X, B=10)

#Compare predicted groups to true groups:
cbind(res$lab.hatK, groups)

#Plot the total within-cluster dispersion and the gap-curve +/- standard errors:
par(mfrow=c(2,1))
plot(1:length(res$W), res$W, type="b")
plot(1:length(res$gap), res$gap, type="b", ylim=c(min(res$gap-res$sk),
max(res$gap+res$sk)), pch=19)
points(1:length(res$sk), res$gap+res$sk, cex=0.7, pch=8)
points(1:length(res$sk), res$gap-res$sk, cex=0.7, pch=8)
segments(x0=1:length(res$sk), y0=res$gap-res$sk, y1=res$gap+res$sk)
```

part *Partitioning Algorithm based on Recursive Thresholding*

Description

The PART method estimates the number of clusters in a data set. It is based on recursive application of the Gap statistic and is able to discover both top-level clusters as well as subclusters nested within the main clusters.

Usage

```
part(X,Kmax=10,minSize=8,minDist=NULL,cl.lab=NULL,...)
```

Arguments

`X`	a numeric data matrix whose rows are to be clustered using a specified clustering algorithm (default is hierarchical clustering with average linkage and Euclidean distance, see below for other options).
`Kmax`	the maximum number of clusters to be evaluated in the first (global) run.
`minSize`	the minimum number of objects required in each cluster.
`minDist`	optional stopping threshold indicating the minimum distance required between two tentative clusters considered for a tentative split. If unspecified the minimum distance is determined by the value of `q`, see below.
`cl.lab`	optional list of length `Kmax` giving vectors of cluster labels for the rows in `X` when partitioned into `1,..,Kmax` clusters.
`...`	other optional parameters. These include the parameters `B` (default 100) and `ref.gen` (default "PC") to be passed on to `gap`, as well as:

- `q`**:** the fraction of dendrogram heights (from the top) used to determine the stopping threshold; only applied if `minDist` is unspecified. Default is 0.25, set `q=1` if no stopping threshold should be applied.
- `Kmax.rec`**:** the maximum number of clusters to consider in each recursive run. Default is 5.
- `cl.method`**:** the desired clustering method. Options currently include "hclust" (default) and "kmeans".
- `linkage`**:** the desired linkage to be applied if `cl.method="hclust"`. Default is "average", see the parameter `method` in `hclust` for other options.
- `dist.method`**:** the desired distance measure to be applied if `cl.method="hclust"`. Default is "euclidean". Other options include those supported by `dist` (under `method`), "sq.euclidean" (squared Euclidean distance) and "cor" (1 minus correlation distance).
- `cor.method`**:** the correlation measure to be used if `dist.method="cor"`. Default is "pearson", see the parameter `method` in `cor` for other options.
- `nstart`**:** the number of initial center sets to be applied if `cl.method="kmeans"`. Default is 10. See `kmeans` for details on this.

Details

PART applies the Gap statistic (Tibshirani et al., 2001) to obtain a global estimate of the number of clusters. If more than one cluster is found, the Gap statistic is re-optimized on each subset of cases corresponding to a cluster. If only one cluster is found, a tentative binary split is made and the objective function is re-optimized on the two tentative clusters. The procedure is repeated recursively until a stopping threshold is reached or the subset under evaluation has less than `2*minSize` cases. Significant clusters (those discovered by Gap) are returned; a tentative cluster is only returned if significant sub-clusters were found solely in the other branch of the tentative split. See Nilsen et al. (2013, preprint) for more details.

Value

`hatK`	the best number of clusters according PART.
`lab.hatK`	a vector of same length as the number of rows in `X` assigning a group label to each case (row) in `X` based on the best partition as evaluated by PART.
`outliers`	a vector indicating which objects are classified as outliers by PART. If no objects are classified as outliers it returns the value `NULL`.

Author(s)

Gro Nilsen

References

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

See Also

`gap`, `plotHeatmap`

Examples

```
## Example 1 ##
#Load a simulated data set with 5 clusters
data(exData1)
X = exData1$X
groups1 = exData1$groups

#Run PART (limit the number of reference data sets to decrease computing time):
res <- part(X, B=10)

#Compare predicted groups to true groups in the data set:
cbind(res$lab.hatK, groups1)

## Visualize results ##
#Transpose the data matrix such that samples are shown in columns:
tX <- t(X)
#Cluster rows and columns using the same clustering method as applied in PART:
rowclust = hclust(dist(tX,method="euclidean"),method="average")
```

```
colclust = hclust(dist(t(tX), method="euclidean"),method="average")
#Order data matrix according to order in clustering og plot heatmap:
X2 = tX[rowclust$order, colclust$order]
par(mar=c(0,0,0,0))
plotHeatmap(X2)
#Add column-dendrogram with leaves colored according to the clusters found by PART:
plotTreeCol(clust=colclust,groups=res$lab.hatK[colclust$order])
#Add color-bar to indicate the true clusters in the data set:
plotColorbarCol(groups=groups1[colclust$order])


## Example 2 ##
# Load a simulated data set with 4 clusters:
data(exData2)
Y = exData2$Y
groups2 = exData2$groups

# Run PART with default clustering method:
res2 = part(Y, B=10)

# Compare predicted groups to true groups in the data set:
cbind(res2$lab.hatK, groups2)

# Visualize results
# Cluster rows and columns using the same clustering method as applied in PART:
rowclust = hclust(dist(Y,method="euclidean"),method="average")
colclust = hclust(dist(t(Y), method="euclidean"),method="average")
# Order data matrix according to order in clustering og plot heatmap:
Y2 = Y[rowclust$order, colclust$order]
par(mar=c(0,0,0,0))
heat = plotHeatmap(Y2)
# Add row-dendrogram with leaves colored according to the clusters found by PART:
plotTreeRow(clust=rowclust,groups=res2$lab.hatK[rowclust$order])
# Add column-dendrogram:
plotTreeCol(clust=colclust)
#Add color-bar to show the true group memberships:
plotColorbarRow(groups=groups2[rowclust$order])


## Some examples showing how to change clustering method and distance measure ##

#Run PART with complete linkage:
res3 <- part(Y, B=10, linkage="complete")

#Run PART with 1 - Pearson correlation distance
res4 <- part(Y, B=10, dist.method="cor")

#Run PART with 1 minus Spearman correlation distance:
res5 <- part(Y, B=10, dist.method="cor", cor.method="spearman")
```

plotColorbarCol *Add a colorbar above a heatmap.*

Description

Plots a color bar above a heatmap created with `plotHeatmap` to annotate the columns.

Usage

```
plotColorbarCol(groups, margin)
```

Arguments

`groups`	optional vector of the same length as the number of columns in the heatmap, and with values specifying the colors of the leaves in the dendrogram.
`margin`	optional vector of length 2 specifying the margins around the heatmap. The first component specifies the width of the inner margin (default value is 0.05) and the second component specifies the width of the outer margin (default value is 0.4). The inner margin surrounds the heatmap on all sides, while the outer margin is only present on the left side and top side of the heatmap. See details in help file for `plotHeatmap` for explanation on how to use this parameter.

Details

This method is designed to be used in conjunction with the method `plotHeatmap`. The argument value `margin` should match the one used in the call to `plotHeatmap`; for details on how to use this argument, see the help file for that method. The default values are the same as in `plotHeatmap`, hence if no `margin` is specified in the call to `plotHeatmap`, then no `margin` need to be specified in the call to `plotColorbarCol` either.

Author(s)

Ole Christian Lingjaerde

References

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

See Also

`plotHeatmap`, `plotColorbarRow`

Examples

```
## Create a dummy data set with 50 rows and 100 columns
X = matrix(rnorm(50*100), 50)
colgroups = sample(1:3, 100, replace=TRUE)

## Plot a heatmap and then add a color bar on the left side
margin = c(0.1, 0)
plotHeatmap(X, margin)
plotColorbarCol(colgroups, margin)
```

`plotColorbarRow` *Add a colorbar on the left side of a heatmap.*

Description

Plots a color bar on the left side of a heatmap created with `plotHeatmap` to annotate the rows.

Usage

```
plotColorbarRow(groups, margin)
```

Arguments

`groups`	optional vector of the same length as the number of rows in the heatmap, and with values specifying the colors of the leaves in the dendrogram.
`margin`	optional vector of length 2 specifying the margins around the heatmap. The first component specifies the width of the inner margin (default value is 0.05) and the second component specifies the width of the outer margin (default value is 0.4). The inner margin surrounds the heatmap on all sides, while the outer margin is only present on the left side and top side of the heatmap. See details in help file for `plotHeatmap` for explanation on how to use this parameter.

Details

This method is designed to be used in conjunction with the method `plotHeatmap`. The argument value `margin` should match the one used in the call to `plotHeatmap`; for details on how to use this argument, see the help file for that method. The default values are the same as in `plotHeatmap`, hence if no `margin` is specified in the call to `plotHeatmap`, then no `margin` need to be specified in the call to `plotColorbarRow` either.

Author(s)

Ole Christian Lingjaerde

References

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

See Also

plotHeatmap, plotColorbarCol

Examples

```
## Create a dummy data set with 50 rows and 100 columns
X = matrix(rnorm(50*100), 50)
rowgroups = sample(1:3, 50, replace=TRUE)

## Plot a heatmap and then add a color bar on the left side
margin = c(0.1, 0)
plotHeatmap(X, margin)
plotColorbarRow(rowgroups, margin)
```

---

plotColorRange *Plot color scale.*

Description

Plots the color scale used by the method `plotHeatmap`.

Usage

```
plotColorRange(colrange)
```

Arguments

`colrange`	output from call to `plotHeatmap`.

Details

This method is designed to be used in conjunction with `plotHeatmap` and plots the color range used in a call to the latter method. It does not create a new graphics device if one is already open, so it is typically preceded by a call to `windows()` or a similar method.

Author(s)

Ole Christian Lingjaerde

References

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

See Also

plotHeatmap

Examples

```
## Create a dummy data set with 50 rows and 100 columns
X = matrix(rnorm(50*100), 50)

## Create a screen with 3 x 3 virtual panels and use the last two rows for the
## heatmap
layout(matrix(c(2,3,3,1,1,1,1,1,1),3,3,byrow=TRUE))
par(mar=c(2,2,2,2))

## Plot heatmap and then the color range above the heatmap
crange = plotHeatmap(X, c(0,0))  # c(0,0) specifies no margin around heatmap
plotColorRange(crange)
```

`plotHeatmap` *Plot a heatmap*

Description

Displays a matrix as a heatmap, i.e. as a grid of colored rectangles with colors corresponding to the values in the matrix.

Usage

```
plotHeatmap(X, margin, fast=FALSE, colorscale="green-black-red",
               pch=".", cex=10)
```

Arguments

`X`	a numerical matrix.
`margin`	optional vector of length 2 specifying the margins around the heatmap. The first component specifies the width of the inner margin (default value is 0.05) and the second component specifies the width of the outer margin (default value is 0.4). The inner margin surrounds the heatmap on all sides, while the outer margin is only present on the left side and top side of the heatmap. See details below for explanation on how to use this parameter.
`fast`	logical value indicating whether to use a fast method or not to plot the heatmap. The fast method plots the entire heatmap with a single call to `plot.xy` and renders individual matrix values as points with shape and size determined by the parameters `pch` and `cex` (see below). The default method plots the heatmap as a sequence of polygons and while more visually pleasing may be slow when the matrix is large.
`colorscale`	color scale used to represent the numerical values in `X`. The name of the color scale consists of three colors separated by hyphens; these represent the left, middle and right range of the numerical scale. The numerical scale is always symmetric around zero, and the three colors represent the values `-max(abs(X))`, `0` and `max(abs(X))`, respectively. Other options are "red-black-green", "green-white-red", "red-white-green" and "blue-white-red".

pch	plot character used to display a rectangle in the heatmap (only used if `fast=TRUE`).
cex	plot character size (only used if `fast=TRUE`).

Details

Displays a numerical matrix as an pseudo-color heatmap where rows and columns in the heatmap correspond to rows and columns in the matrix, respectively. The lower left corner of the heatmap corresponds to the first element `X[1,1]` of the matrix. The parameter `margin` is used to specify appropriate margins around the heatmap when this is to be combined with color bars and/or cluster dendrograms. Color bars (produced with the functions `plotColorbarRow` and `plotColorbarCol`) are plotted in the inner margin area, while cluster dendrograms (produced with the functions `plotTreeRow` and `plotTreeCol`) are plotted in the outer margin area. Both margins should be specified as a number between 0 and 1, and to leave no margins around the heatmap let `margin=c(0,0)`.

Value

A list with three elements `colorscale`, `value.range` and `color.range`. The list is mainly intended to be used as input for the function `plotColorRange`.

Author(s)

Ole Christian Lingjaerde

References

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

See Also

`plotColorbarCol`, `plotColorbarRow`, `plotTreeCol`, `plotTreeRow`, `plotColorRange`.

Examples

```
## Create a data matrix
A = cbind(matrix(rnorm(70*20),nrow=70), matrix(3+rnorm(70*20),nrow=70))
B = cbind(-1 + matrix(rnorm(40*30),nrow=40), matrix(1+rnorm(40*10),nrow=40))
X = rbind(A,B)[sample(110), sample(40)]

## Cluster rows and columns
rowclust = hclust(dist(X, method="euclidean"), method="complete")
colclust = hclust(dist(t(X), method="euclidean"), method="average")

## Plot data in original order and in clustered order
par(mfrow=c(1,2))
plotHeatmap(X, c(0,0))
plotHeatmap(X[rowclust$order, colclust$order])
plotTreeRow(rowclust)
plotTreeCol(colclust)

## Identify groups with PART and color the leaves of the dendrogram accordingly
rowgroups = part(X, B=10, linkage="complete")$lab.hatK
```

```
colgroups = part(t(X), B=10)$lab.hatK
plotHeatmap(X[rowclust$order, colclust$order])
plotTreeRow(rowclust, rowgroups[rowclust$order])
plotTreeCol(colclust, colgroups[colclust$order])
```

`plotTreeCol` *Add a column dendrogram to a heatmap.*

Description

Plots a column dendrogram above a heatmap created with `plotHeatmap`.

Usage

```
plotTreeCol(clust, groups, margin)
```

Arguments

`clust`	output from call to `hclust` to cluster the columns in the heatmap.
`groups`	optional vector of the same length as the number of columns in the heatmap, and with values specifying the colors of the leaves in the dendrogram.
`margin`	optional vector of length 2 specifying the margins around the heatmap. The first component specifies the width of the inner margin (default value is 0.05) and the second component specifies the width of the outer margin (default value is 0.4). The inner margin surrounds the heatmap on all sides, while the outer margin is only present on the left side and top side of the heatmap. See details in help file for `plotHeatmap` for explanation on how to use this parameter.

Details

This method is designed to be used in conjunction with the method `plotHeatmap`. The argument value `margin` should match the one used in the call to `plotHeatmap`; for details on how to use this argument, see the help file for that method. The default values are the same as in `plotHeatmap`, hence if no `margin` is specified in the call to `plotHeatmap`, then no `margin` needs to be specified in the call to `plotTreeCol` either.

Author(s)

Ole Christian Lingjaerde

References

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

See Also

`plotHeatmap`, `plotColorbarCol`, `plotColorbarRow`, `plotTreeRow`, `plotColorRange`

Examples

```
## See example in the help file for plotHeatmap.
```

`plotTreeRow` *Add a row dendrogram to a heatmap.*

Description

Plots a row dendrogram to the left of a heatmap created with `plotHeatmap`.

Usage

```
plotTreeRow(clust, groups, margin)
```

Arguments

`clust`	output from call to `hclust` to cluster the rows in the heatmap.
`groups`	optional vector of the same length as the number of rows in the heatmap, and with values specifying the colors of the leaves in the dendrogram.
`margin`	optional vector of length 2 specifying the margins around the heatmap. The first component specifies the width of the inner margin (default value is 0.05) and the second component specifies the width of the outer margin (default value is 0.4). The inner margin surrounds the heatmap on all sides, while the outer margin is only present on the left side and top side of the heatmap. See details in help file for `plotHeatmap` for explanation on how to use this parameter.

Details

This method is designed to be used in conjunction with the method `plotHeatmap`. The argument value `margin` should match the one used in the call to `plotHeatmap`; for details on how to use this argument, see the help file for that method. The default values are the same as in `plotHeatmap`, hence if no `margin` is specified in the call to `plotHeatmap`, then no `margin` needs to be specified in the call to `plotTreeRow` either.

Author(s)

Ole Christian Lingjaerde

References

Nilsen et al., "Identifying clusters in genomics data by recursive partitioning", 2013 (in review)

See Also

`plotHeatmap`, `plotColorbarCol`, `plotColorbarRow`, `plotTreeCol`, `plotColorRange`

Examples

```
## See example in the help file for plotHeatmap.
```

Index