Package ‘iterClust’

May 18, 2018

Type Package
Title Iterative Clustering
Version 1.2.0
Author Hongxu Ding and Andrea Califano
Maintainer Hongxu Ding <hd2326@columbia.edu>
Description A framework for performing clustering analysis iteratively.
License file LICENSE
Depends R (>= 3.4.1)
LazyData TRUE
Imports Biobase, cluster, stats, methods
Suggests tsne, bcellViper
biocViews StatisticalMethod, Clustering
URL https://github.com/hd2326/iterClust
BugReports https://github.com/hd2326/iterClust/issues
RoxygenNote 6.0.1

R topics documented:
clustEval .......................................................... 2
clustHetero ......................................................... 2
coreClust .......................................................... 3
featureSelect ....................................................... 4
iterClust .......................................................... 4
obsEval ............................................................ 6
obsOutlier ........................................................ 7

Index 9
**clustEval**  
*Cluster-wise Clustering Robustness Evaluation*

**Description**
A sample cluster-wise clustering robustness evaluation framework (described in "Examples" section, used as default in iterClust framework). Customized frameworks can be defined following rules specified in "Usage", "Arguments" and "Value" sections.

**Usage**
```r
clustEval(dset, iteration, clust)
```

**Arguments**
- **dset** (numeric matrix) features in rows and observations in columns
- **iteration** (positive integer) specifies current iteration
- **clust** return value of coreClust

**Value**
a numeric vector, specifies the clustering robustness (higher value means more robust) of each clustering scheme

**Author(s)**
DING, HONGXU (hd2326@columbia.edu)

**Examples**
```r
clustEval <- function(dset, iteration, clust){
  dist <- as.dist(1 - cor(dset))
  clustEval <- vector("numeric", length(clust))
  for (i in 1:length(clust)){
    clustEval[i] <- mean(silhouette(clust[[i]], dist)[, "sil_width"])
  }
  return(clustEval)
}
```

**clustHetero**  
*Cluster Heterogeneity Evaluation*

**Description**
A sample cluster heterogeneity evaluation framework (described in "Examples" section, used as default in iterClust framework). Customized frameworks can be defined following rules specified in "Usage", "Arguments" and "Value" sections.

**Usage**
```r
clustHetero(clustEval, iteration)
```
Arguments

clustEval, return value of clustEval
iteration (positive integer) specifies current iteration

Value

a boolean vector, specifies whether clusters are heterogenous

Author(s)

DING, HONGXU (hd2326@columbia.edu)

Examples

clustHetero <- function(clustEval, iteration){
  return(clustEval > 0*iteration+0.15})

coreClust <- function(dset, iteration){
  dist <- as.dist(1 - cor(dset))
  range=seq(2, ncol(dset)-1, by = 1)
  clust <- vector("list", length(range))
  for (i in 1:length(range)) clust[[i]] <- pam(dist, range[i])$clustering
  return(clust)}
featureSelect  

**Feature Selection**

**Description**

A sample feature selection framework (described in "Examples" section, used as default in `iterClust` framework). Customized frameworks can be defined following rules specified in "Usage", "Arguments" and "Value" sections.

**Usage**

```r
featureSelect(dset, iteration, feature)
```

**Arguments**

- **dset** (numeric matrix) features in rows and observations in columns
- **iteration** (positive integer) specifies current iteration
- **feature** (character array) specifies user defined features, facilitating feature selection

**Value**

a character array, contains features selected

**Author(s)**

DING, HONGXU (hd2326@columbia.edu)

**Examples**

```r
featureSelect <- function(dset, iteration, feature) return(rownames(dset))
```

iterClust  

**Iterative Clustering**

**Description**

A framework for performing clustering analysis iteratively

**Usage**

```r
iterClust(dset, maxIter = 10, minFeatureSize = 100,
          featureSelect = iterClust::featureSelect, minClustSize = 10,
          coreClust = iterClust::coreClust, clustEval = iterClust::clustEval,
          clustHetero = iterClust::clustHetero, obsEval = iterClust::obsEval,
          obsOutlier = iterClust::obsOutlier)
```
Arguments

dset          (numeric matrix or data.frame) features in rows and observations in columns, or
SummarizedExperiment0 and ExpressionSet object
maxIter       (positive integer) specifies maximum number iterations to be performed
minFeatureSize (positive integer) specifies minimum number of features needed
featureSelect (function) takes a dataset, depth(IV) and cluster$feature(IV), returns a character
array, containing features used for clustering analysis
minClustSize  (positive integer) specifies minimum cluster size
coreClust     (function) takes a dataset and depth(IV), returns a list, containing clustering
vectors under different clustering parameters
clustEval     (function) takes a dataset, depth(IV) and coreClust result, returns a numeric vector,
evaluating the robustness (higher value means more robust) of each clustering scheme
clustHetero   (function) takes depth(IV) and clustEval result, returns a boolean vector, deciding
whether a cluster is considered as heterogenous
obsEval       (function) takes a dataset and optimal coreClust result determined by clustEval,
returns a numeric vector, evaluating the clustering robustness of each observation
obsOutlier    (function) takes depth(IV) and obsEval result, returns a boolean vector, deciding
whether an observation is outlier

Details

######################################################################## General Idea ####################################################################
In a scenario where populations A, B1, B2 exist, pronounce differences between A and B may mask
subtle differences between B1 and B2. To solve this problem, so that heterogeneity can be better
detected, clustering analysis needs to be performed iteratively, so that, for example, in iteration 1,
A and B are seperated and in iteration 2, B1 and B2 are seperated.

######################################################################## General Work Flow ####################################################################
iith Iteration Start ==»
featureSelect (feature selection) ==»
minFeatureSize (confirm enough features are selected) ==»
clustHetero (confirm heterogeneity) ==»
coreClust (generate several clustering schemes to be evaluated) ==»
clustEval (pick optimal clustering scheme generated in previous step) ==»
minClustSize (remove clusters with few observations) ==»
obsEval (evaluate how each observations are clustered) ==»
obsOutlier (remove poorly clustered observations) ==»
results in Internal Variables (IV) ==»
iith Iteration End

######################################################################## Internal Variables (IV) ####################################################################
The following IVs are used in user-defined functions in each iteration:
cluster: (list) the return value, described in "Value" section
depth: (numeric) current round of iteration
Value

a list with the following structure containing iterClust result

→ $cluster (list) $Iter[i] (list) $Cluster[j], (character array) names of observations belong to each cluster

→ $feature (list) $Iter[i] (list) $Cluster[j], (character array) features used to split each cluster in the previous iteration thereby produce the current clusters

→ $clusterScore (list) $Iter[i] (list) $Cluster[j], (numeric array) clustEval output for each clustering schemes

→ $observationScore (list) $Iter[i] (list) $Cluster[j], (numeric array) obsEval output for each samples

Author(s)

DING, HONGXU (hd2326@columbia.edu)

Examples

library(tsne)
library(cluster)
library(bcellViper)

data(bcellViper)
exp <- exprs(dset)
pheno <- as.character(dset@phenoData@data$description)
exp <- exp[, pheno %in% names(table(pheno))[,table(pheno) > 5]]
pheno <- pheno[pheno %in% names(table(pheno))[,table(pheno) > 5]]

#load bcellViper expression and phenotype annotation

c <- iterClust(exp, maxIter=3, minClustSize=5)

dist <- as.dist(1 - cor(exp))
set.seed(1)

for (j in 1:length(c$cluster)){
  COL <- structure(rep(1, ncol(exp)), names = colnames(exp))
  for (i in 1:length(c$Cluster[[j]]) - 1)
    plot(tsne[, 1], tsne[, 2], cex = 0, cex.lab = 1.5,
         xlab = "Dim1", ylab = "Dim2",
         main = paste("iterClust, iter=" , j, sep = ""))
    text(tsne[, 1], tsne[, 2], labels = pheno, cex = 0.5, col = COL)
    legend("topleft", legend = "Outliers", fill = 1, bty = "n")
  #visualize results
**obsOutlier**

**Usage**

```r
obsEval(dset, clust, iteration)
```

**Arguments**

- `dset` (numeric matrix) features in rows and observations in columns
- `clust` optimal return value of coreClust
- `iteration` (positive integer) specifies current iteration

**Value**

a numeric vector, specifies the clustering robustness (higher value means more robust) of each observation under the optimal clustering scheme

**Author(s)**

DING, HONGXU (hd2326@columbia.edu)

**Examples**

```r
obsEval <- function(dset, clust, iteration){
  dist <- as.dist(1 - cor(dset))
  obsEval <- vector("numeric", length(clust))
  return(silhouette(clust, dist)[, "sil_width"])
}
```

---

**obsOutlier**

*Outlier Observation Evaluation*

**Description**

A sample outlier observation evaluation framework (described in "Examples" section, used as default in iterClust framework). Customized frameworks can be defined following rules specified in "Usage", "Arguments" and "Value" sections.

**Usage**

```r
obsOutlier(obsEval, iteration)
```

**Arguments**

- `obsEval`, return value of obsEval
- `iteration` (positive integer) specifies current iteration

**Value**

a boolean vector, specifies whether an observation is outlier

**Author(s)**

DING, HONGXU (hd2326@columbia.edu)
Examples

obsOutlier <- function(obsEval, iteration) return(obsEval < 0*iteration-1)
Index

*Topic clustEval
clustEval, 2

*Topic clustHetero
clustHetero, 2

*Topic coreClust
coreClust, 3

*Topic featureSelect
featureSelect, 4

*Topic iterClust
iterClust, 4

*Topic obsEval
obsEval, 6

*Topic obsOutlier
obsOutlier, 7